

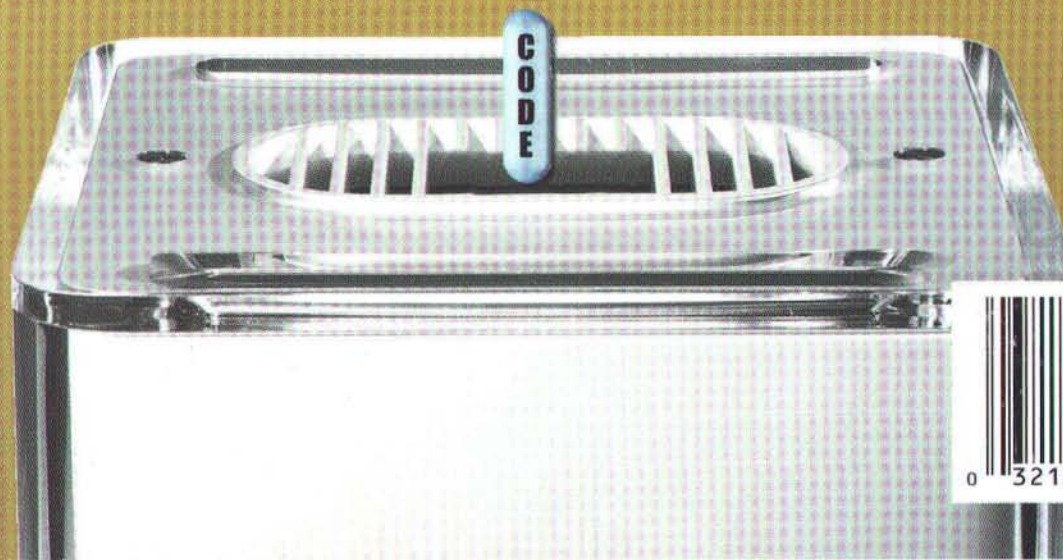
MacTech®

The Journal of Macintosh Technology and Development

```
> var myObject = { 'itemType' : 'auto',  
                  > 'subtype' : '4WD',  
                  > 'make' : 'Jeep',  
                  > 'year' : '1998',  
                  > 'color' : 'green' };  
                  >  
                  > var myObject = new Object();  
                  > myObject['itemType'] = 'auto';  
                  > myObject['subtype'] = '4WD';  
                  > myObject['make'] = 'Jeep';  
                  > myObject['year'] = '1998';  
                  > myObject['color'] = 'green';
```

How to Harness the Power of Associative Arrays

By Kas Thomas



\$8.95 US
\$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.





Keeping Mac dreams alive since 1993.

**Now
Shipping!**
CodeWarrior
for Mac OS &
Mac OSx

Two great operating systems,
one CodeWarrior.

Mac developers depended on CodeWarrior to make the platform architecture shift from 68K to PowerPC processors. Now CodeWarrior does it again, speeding your transition to the next new operating system. CodeWarrior for Mac OS, Version 6.0 supports development for both OS X

and Classic Mac operating systems from a single, powerful, award-winning Integrated Development Environment.

Discover how CodeWarrior for Mac OS, Version 6.0 can help you realize your Mac development dreams.

Visit www.metrowerks.com/go/mac.

CodeWarrior®



**"Without a doubt, the Premiere Resource Editor
for the Mac OS ... A wealth of time-saving tools."**

— MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."

— MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."

— Leonard Rosenthol, Aladdin Systems

**"Without Resorcerer, our localization efforts would look like a
Tower of Babel. Don't do product without it!"**

— Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."

— Bill Goodman, author of *Smaller Installer* and *Compact Pro*

"Resorcerer Rocks! Buy it, you will NOT regret it."

— Joe Zobkiw, author of *A Fragment of Your Imagination*

"Resorcerer will pay for itself many times over in saved time and effort."

— MacUser review

"The template that disassembles 'PICT's is awesome!"

— Bill Steinberg, author of *Pyro!* and *PBTools*

"Resorcerer proved indispensable in its own creation!"

— Doug McKenna, author of *Resorcerer*



Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater,
1.5MB RAM, CD-ROM

Standard price: \$256 (decimal)

Website price: \$128 - \$256

(Educational, quantity, or
other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMAESTHETICS, INC.

PO Box 298

Boulder, CO 80306-0298 USA

Phone: (303) 440-0707

Fax: (303) 440-0504

resorcerer@mathemaesthetics.com

New
in
2.0:

- Very fast, HFS browser for viewing file tree of all volumes
- Extensibility for new Resorcerer Apprentices (CFM plug-ins)
- New AppleScript Dictionary ('aete') Apprentice Editor
- MacOS 8 Appearance Manager-savvy Control Editor
- PowerPlant text traits and menu command support
- Complete AIFF sound file disassembly template
- Big-, little-, and even mixed-endian template parsing
- Auto-backup during file saves; folder attribute editing
- Ships with PowerPC native, fat, and 68K versions

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'ictb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail**?

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS

Orders, Circulation, & Customer Service

Press Releases

Ad Sales

Editorial

Programmer's Challenge

Online Support

Accounting

Marketing

General

Web Site (articles, info, URLs and more...)

E-Mail/URL

cust_service@mactech.com

press_releases@mactech.com

ad_sales@mactech.com

editorial@mactech.com

prog_challenge@mactech.com

online@mactech.com

accounting@mactech.com

marketing@mactech.com

info@mactech.com

http://www.mactech.com

The MacTech Editorial Staff

Publisher • Neil Ticktin

Managing Editor • Jessica Stubblefield

Online Editor • Jeff Clites

Regular Columnists

Networking

by John C. Welch

Programmer's Challenge

by Bob Boonstra

MacTech Online

by Jeff Clites

From the Factory Floor

by Metrowerks

Tips & Tidbits

by Jeff Clites

Regular Contributors

Vicki Brown, Andrew Stone,
Tim Monroe, Erick Tejkowski,
Kas Thomas, Will Porter,
Paul E. Seving, Tom Djajadiningrat,
and Jordan Dea-Mattson

MacTech's Board of Advisors

Dave Mark, Jordan Dea-Mattson,
Jim Straus, Jon Wiederspan,
and Eric Gundrum

MacTech's Contributing Editors

- Jim Black, Apple Computer, Inc.
- Michael Brian Bentley
- Tantek Çelik, Microsoft Corporation
- Marshall Clow
- John C. Daub
- Tom Djajadiningrat
- Bill Doerrfeld, Blueworld
- Andrew S. Downs
- Richard V. Ford, Packeteer
- Gordon Garb, Cobalt Networks
- John Hanay, Apple Computer
- Lorca Hanns, San Francisco State University
- Ilene Hoffman
- Chris Kilbourn, Digital Forest
- Scott Knaster, Microsoft
- Mark Kriegsman, Clearway Technologies
- Peter N. Lewis, Stairways Software
- Bill McGlasson, Apple Computer
- Rich Morin
- Terje Norderhaug, Media*Design-In-Progress
- Nathan Nunn, Purity Software
- John O'Fallon, Maxum Development
- Alan Oppenheimer, Open Door Networks
- Avi Rappoport, Search Tools Consulting
- Ty Shipman, Kagi
- Chuck Shotton, BIAP Systems
- Cal Simone, Main Event Software
- Steve Sisak, Codewell Corporation
- Dori Smith
- Andrew C. Stone, www.stone.com
- Michael Swan, Neon Software
- Chuck Von Rospach, Plaidworks
- Bill von Hagen
- Eric Zelenka

Xplain Corporation Staff

Chief Executive Officer • Neil Ticktin

President • Andrea J. Sniderman

Controller • Michael Friedman

Production Manager • Jessica Stubblefield

Production/Layout • W2 Graphics

Marketing Managers

Nick DeMello and Alyse Yourist

Events Manager • Susan M. Worley

Network Administrator • Steve Ruge

Accounting • Jan Webber, Marcie Moriarty

Customer Relations • Laura Lane

Susan Pomrantz

Shipping/Receiving • Joel Licardie

Board of Advisors • Steven Geller, Blake Park,
and Alan Carsrud

All contents are Copyright 1984-2000 by Xplain Corporation. All rights reserved. MacTech, Developer Depot, and Sprocket are registered trademarks of Xplain Corporation. Depot, The Depot, Depot Store, Video Depot, MacDev-1, THINK Reference, NetProfessional, NetProLive, JavaTech, WebTech, BeTech, ExplainIt, and the MacTutorMan are trademarks of Xplain Corporation. Other trademarks and copyrights appearing in this printing or software remain the property of their respective holders. Xplain Corporation does not assume any liability for errors or omissions by any of the advertisers, in advertising content, editorial, or other content found herein. Opinions or expressions stated herein are not necessarily those of the publisher and therefore, publisher assumes no liability in regards to said statements.



This publication is
printed on paper with
recycled content.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are \$47.00 per year. Canadian subscriptions are \$59.00 per year. All other international subscriptions are \$97.00 per year. Domestic source code disk subscriptions are \$77 per year. All international disk subscriptions are \$97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

C o n t e n t s

October 2000 • Volume 16, Issue 10

Feature Articles

14 QUICKTIME TOOLKIT

14 Somewhere I'll Find You

by Tim Monroe

34 WEB OBJECTS

34 How to Build an EOModel

by Patrick Taylor and Sam Krishna

70 MAC OS X

70 An Intro to Mac OS X's Command Line Interface

by Andrew C. Stone

CARBON DEVELOPMENT

48 A Select Few...

by Daniel Jalkut

56 WEB OBJECTS

56 WebObjects Development Lifecycle

by David K. Every

82 TOOLS OF THE TRADE

82 Satimage's Smile

by Tom Djajadiningrat

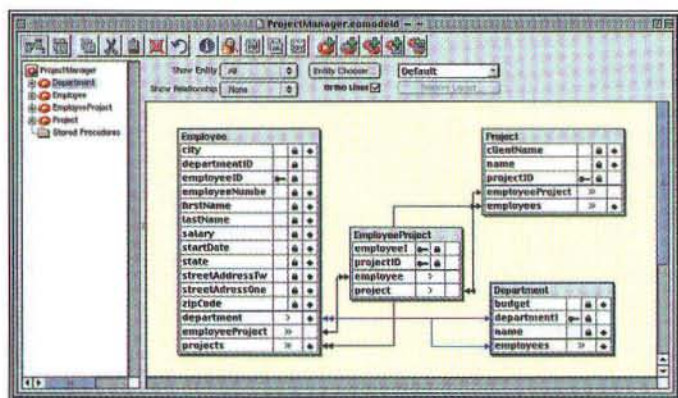
98 Copyleft or Copyright

by Stephen Fantl

102 OPINION

102 Cable Modem Guide

by Ilene Hoffman

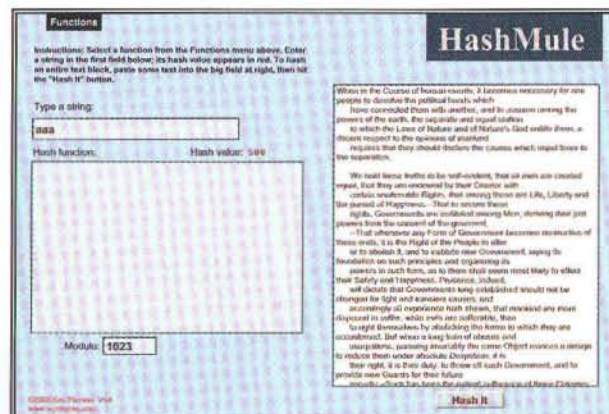


WebObjects.....Page 34

COVER STORY

40 How to Harness the Power of Associative Arrays

by Kas Thomas



Associative Arrays.....Page 40

Columns

4 VIEWPOINT

by Neil Ticktin

GETTING STARTED NETWORKING

6 Networks 201 Pt. 4

by John C. Welch

PROGRAMMER'S CHALLENGE

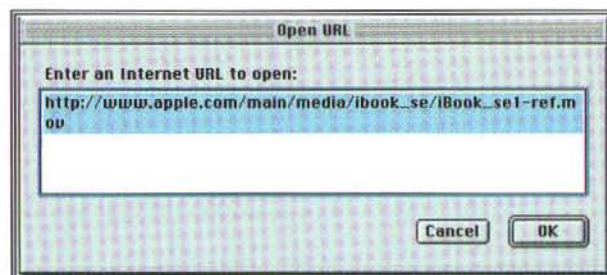
90 What Bills Did They Pay?

by Bob Boonstra

MACTECH ONLINE

94 CVS: Version Control is your friend

by Jeff Clites



QuickTime Toolkit.....Page 14

By Neil Ticktin, Publisher, MacTech Magazine

MAC OS X... ARE WE THERE YET?

I'm sit here today, having just received my copy of Mac OS X beta, and I begin to wonder, as you probably are... what's going to happen with this? When will developers make the jump to X? What does the future hold for us as developers? Now, while I won't profess to being a soothsayer, I'll take the plunge and put my opinion in print for the world to see.

Apple is Serious... this time

The developer community has been burned by Apple's promise of a modern operating system since, say, the early to mid-1990's. Yes, it has been that long. We've been through the original System 8 (not the one that shipped), Copland, rumors of the BeOS, Rhapsody, Yellow Box, and more. Time and time again, Apple has simply not delivered the promise of a "modern operating system". Some were technical problems, some were business problems... and some were just plain old change in direction.

So, why should we believe that this time Apple is going to make this happen?

Two reasons: Steve Jobs and Public Beta. Steve Jobs can't stand to be publicly humiliated. He's stood up saying, "the train has left the station", and that (paraphrased) we're moving forward with Mac OS X, not as a multi-prong strategy, but as a single strategy for the future. Now, whether they ship Q1 2000 or Q2, I don't know... and it doesn't matter in the long run. Steve Jobs will make sure it ships in one form or another, relatively early on next year. He just won't stand for it being otherwise.

How do we know that this isn't just some promise as we've heard before? A public beta. Not only does the general public have tangible evidence in its hands, but early reports are that it runs fairly well... a good indication that we're, in relative terms, close.

Of course, there's always a third reason why things are different this time. Apple has moved into the arena of other, very profitable OS vendors... they are charging for a beta! :) But seriously, you have to believe that they will follow through this time since they are bold enough to charge for the beta.

YEAH, YEAH... I'LL WAIT

There are already a lot of developers moving to X. But, some of you are holding back. This may be an important mistake, but not for the reasons that you are thinking.

It's pretty clear that the Classic environment on Mac OS X (i.e., the environment responsible for running non-carbonized Mac OS 9 applications) is going to work well. So, yes, your customers can run their Classic apps under X.

The issue is that launching the Classic environment takes time... and frankly, even if it can do something about it, Apple has no incentive to make it launch faster. Your customers will quickly learn that they want to avoid launching the Classic environment if they have a choice.

That's where you come in... you can make their experience with your product a good one, or you can set yourself up for them being disappointed in the Classic situation and blaming your product.

Bottom line: customers will want native and carbonized apps so they can avoid Classic. Now is the time to start working on it.

BUT I HAVE SO MUCH TO DO!

If you are like me, you have a million things to do, and not enough time to do them. Working on a new version of your application for an operating system that is technically not shipping, just hasn't bubbled up on the list. So, why should you move forward today?

First, I'm not going to tell you what you should do with your project... that's your call to make. Only you have the information to make such a decision. Though, there are a few things to consider.

Anecdotally at least, carbonizing is relatively easy. For example, a typical app could easily take as little as 2 weeks to carbonize to get it up and running. A bit longer to iron the issues out, and then of course testing. But, this is not a several month project, as it may seem.

More importantly, there are other benefits to carbonizing. Let me explain.

The things that slow folks down the most when carbonizing their apps typically are their own bugs. Many engineers are thrilled because they are re-writing bad code that they've wanted to get to for a while, but haven't had time.

Writing code that is properly carbonized actually turns out to be a good way to debug and test your program — and may result in lower QA or tech support costs. Mac OS X tells you now that you are stepping on memory... not some day down the road when you aren't expecting it. In addition, the development and debugging turn around time is much faster than with Mac OS 9 — in part because, you aren't generally crashing your system as you tracking down bugs.

The bottom line is that carbonizing your app is not the challenge or pain that people think it is. It's actually a lot easier, and once through the process, people are generally happy they did it citing that it wasn't as hard as they thought it would be.

LEADING THE MARKET

Look... Apple is going to push really, really hard here... hard on their own engineers, and hard on developers. The timing of when they will ship 1.0 will be dependent on two things: engineering and market response. If the market responds saying that they want lots of things changed, then Apple will have to do those in addition to the other items they already know about. Rest assured though... a 1.0 version of Mac OS X will ship sooner rather than later.

Furthermore, the acceptance of X as the current (and only) operating system will depend on market acceptance. I expect to see Apple pushing the timeline forward just ahead of the market. They've got experience with this... just the way they dove in on USB, FireWire, and even Wireless Networking. Yes, it will be a bit painful to make such a leap, but in recent years, Apple has been able to keep that pain period relatively short (look at USB).

Bottom line: Jobs is right... the train has left the station. If you are going to support Mac OS X in a way that will help promote your product, the time is now to carbonize.



The key to thinking different...



**Works
on the
iMac!**

The New MacHASP USB Key!

Question: Is MacHASP USB* a software security key or a sales tool?

Answer: It's both!

MachASP USB is the world's first software protection key for the iMac. It gives you sophisticated license enforcement and comprehensive protection against illegal use ... in other words, real security.

Then it gives you a big selling advantage.

With MachASP USB, you can license multiple software modules and applications. You can instantly unlock or upgrade them in the field. Plus you can freely distribute demos.

Bottom line: MachASP USB locks out illegal users and unlocks your

full sales potential – without getting in anyone's way. Call now to request a Developer's Kit and our newly published guide to USB features and benefits.

*For all USB-equipped Macs running an OS with USB support. Fully compatible with the ADB version of MachASP.



Mac OS



USA: 1-800-223-4277
212-564-5678

Int'l: 972-3-6362222

www.aks.com/mt

ALADDIN®
KNOWLEDGE SYSTEMS LTD

Mac software protection provider, Micro Macro Technologies, becomes part of Aladdin, giving its customers even better service from the number one name.

By John C. Welch

Networks 201 pt. 4

Layer 2: The Data Link Layer

REFRESH

Before we start with this month's installment of our look at networks, an apology/error fix. In the last article, we stated that to find the wavelength of signal, you would invert the frequency, via the equation $1/\text{freq}$. Unfortunately, that was incorrect. That equation actually gives you the period of the signal. To get the wavelength, you would more correctly use the equation c/freq , where c represents the speed of light in meters per second. Many thanks to Bruce Toback who was the first to notify us as to the error, and all the other readers who caught it as well. Now on to Layer 2.

Going back to the overview of the OSI model in the first article in the series, Layer 2 is the DataLink Layer, and communicates with Layers 1 and 3. The most basic description of Layer 2's function would be that it receives data and routing information from Layer 3, and assembles them into frames which are passed onto the Layer 1. It also receives serial bitstreams from the Layer 1, and assembles these into frames, which are then passed onto Layer 3. Like most networking functions, the actual duties of this Layer are far more complex, and it is those duties, and the complexity therein that we will look at in this article.

LAYER 2

Base Function

The basic function of the Data Link Layer is to provide services to the

Network Layer. This centers around getting data from the transmitting machine to the receiving machine intact. There are three base methods for doing this:

- 1) Unacknowledged Connectionless Service
- 2) Acknowledged Connectionless Service
- 3) Acknowledged Connection - Oriented Service

The first method, unacknowledged connectionless service, is where the source sends independent frames to the source. This has some analogies to messages in a bottle. You write the message, pop it in the bottle, and set the bottle in the water. It either gets there, or does not. You have no way to verify that it was successfully received, or that, in the case of multiple messages, that they were received in the proper order. This method may sound very unreliable, but in fact it is used quite often. If the protocol you are using, such as TCP, has provisions for connection management, message reassembly, and acknowledgement at a higher layer, then there is no sense in having an additional level of acknowledgement and connection management in the Data Link Layer. As a result, most LANs use this type of service at the Data Link Layer level. The other reason for using this service would be in real time situations, where the time delays in setting up connections and retransmitting data would cause delays that would impede the function of the real time applications.

The second type of service is acknowledged connectionless service. This is analogous to registered mail with a delivery receipt. You have no idea how the mail got to its destination, but you know that it was received or not. There are two ways to deal with acknowledgement errors. The first is to retransmit the entire set of data. While safer in theory, this is impractical for a number of reasons. The first is that if we are talking about a large amount of data, retransmitting the entire data set can take an unacceptably long time, especially over slow links. The other is that considering the chances of a lost packet here and there get

John Welch <jwelch@aer.com> is the Mac and PC Administrator for AER Inc., a weather and atmospheric science company in Cambridge, Mass. He has over fifteen years of experience at making computers work. His specialties are figuring out ways to make the Mac do what nobody thinks it can, and showing that the Mac is the superior administrative platform.

very high, especially over unreliable links, so in theory, you could run into a situation where you would never be able to stop retransmitting your data set. The second, and more common method is to only retransmit those frames that were lost. This method requires more sophisticated checking algorithms, but has a greater efficiency than retransmitting the entire message for every error. While lost frames are not a great issue with reliable media, such as fiber, when we talk about wireless networking, the chances for lost frames are much greater, and in fact, this is the type of service used in the 802.11b wireless networking standard. It is worth noting that providing acknowledgements at the Data Link Layer level is an optimization, not a requirement. That function can be, and often is handled at higher levels.

The final type of service is acknowledged connection-oriented service. In this class of service, a connection is created before any data is transmitted, each frame sent over the connection is numbered, and each frame sent is guaranteed to be received. In addition, each frame sent is guaranteed to be received only once, and in the correct order. This type of service creates, what is essentially a networked bit stream. There are three phases to data transfers in this service. The first is the establishment of a connection. As part of this, frame counters are created on both sides, to keep track of which frames have and have not been sent. The second phase is the actual transmitting of

data, and the tracking of the frames. The final phase is the connection teardown, and the disposal of the frame counters and other resources used.

It worth mentioning here, that as far as the Data Link Layer is concerned, the Physical Layer doesn't really exist. Although the physical path of the data has to travel through the Physical Layer, the logical path at the Layer 2 level would be a direct connection between Layer 2 on the sender and Layer 2 on the receiver. There are a number of things including error correction, data delivery and flow control. An important thing to remember is that for the Physical Layer to really exist

the Physical Layer, the Data Link Layer deals with the Physical Layer. The Physical Layer both provides service to the Data Link Layer and uses the services provided by the Data Link Layer. Remember, all the Physical Layer cares about is getting bits from the Network Layer, and shoving them out onto the line to their destination. (Actually, all the Physical Layer cares about is shoving bits onto and receiving bits from the wire. It has nothing to do with addressing, and in the case of things like Ethernet, will actually look at all bits on the wire, relying on the higher layers to check things like destination addressing.)

Imagine that. Smart and beautiful.

Okay, we admit it.
A lot of plastic is involved.)



Think about it.
Why should you have to buy a USB hub, then hang stuff off of it until it looks like something out of *Edward Scissorhands*? Why fumble around under your desk every time you want to plug in another device? Instead, try this. A modular, stacking system of USB hubs, with adapters for serial and SCSI. Neat. Compact. And, oh yeah... drop dead gorgeous.



belkin.com

Belkin Components
Compton • CA • USA
310.898.1100 • Fax 310.898.1111
United Kingdom • Holland • Atlanta, GA

In any case, the structure used by the Data Link Layer is the frame. All data from the Network Layer is encapsulated into a frame, and sent on to the Physical Layer. Conversely, all bits from the Physical Layer is packed into frames, and sent on to the Network Layer. Although the specific sizes and contents of frames are determined by the hardware protocol used, such as Ethernet and Token Ring, all frames have certain structural commonalities.

All frames have a Start Of Frame delimiter of some kind. This is some kind of structure that says "This is where the frame starts". They also all have some kind of End Of Frame delimiter, or Frame Check Sequence, that says "This is where the frame ends." Since we are talking about the start and end of frame delimiters, this is a good time to touch on exactly how frames are pulled out of bits on a wire. There are about five ways to do this. The first, timing, is not used, and was never really used, as there is almost no way for any network to guarantee the timing between frames, or how long a frame takes to get from point a to point b. This leaves us with four other methods of marking frames:

- 1) Character Count
- 2) Starting and ending characters, with character stuffing
- 3) Starting and ending flags, with bit stuffing.
- 4) Physical Layer coding violations

Method one, character count, uses a header to specify the number of characters in a frame. When the Data Link Layer sees this header, it knows the end of the frame is exactly X number of characters follow the character count header, and therefore, where the frame ends. The problem with this is that there is no real way to deal with the character count getting scrambled. If the transmitter sends a count of 25 characters, and the data gets scrambled, so that the receiver sees a count of 20 characters, then not only is that frame garbled, but all following frames as well. Once this synchronization is lost, then even retransmission doesn't work, because there is no way to tell how many characters to ignore so as to skip the bad frame. Not surprisingly, the character count method is rarely used these days.

The second method uses specific characters to specify the beginning and the end of the frame structure. The characters used are ASCII character pairs, with DLE STX used for the frame start, and DLE ETX used for the frame finish. (DLE stands for Data Link Escape, STX stands for Start of TeXt, and ETX stands for End of TeXt.) By using specific character sets as frame delimiters, and using those specific character pairs solely as frame delimiters, the character count synchronization issues are avoided. This works well for text data, but if we are dealing with binary or numerical data, then it is possible for those characters to occur in random places within the frame. The way to avoid this problem is to have the Data Link Layer insert, or 'stuff', an extra DLE character in front of each occurrence of a delimiter pair in the wrong place. This way, the Data Link Layer on the

receiving end knows that a double DLE pair is not a frame delimiter, and to remove one of the DLE characters from the frame data before passing the frame up to the Network Layer. Although this character stuffing works reasonably well, this entire method is too closely tied to eight-bit ASCII data to be useful universally.

The third method, uses bit pattern flags instead of character pairs as frame delimiters. This flag pattern is generally 01111110. To avoid synchronization errors caused by that pattern occurring naturally, every time there is a series of five consecutive 1 bits, a 0 bit is inserted immediately after the fifth 1. This way, the flag pattern is never duplicated in the frame data, avoiding synchronization errors. As well, this method is not tied to any particular encoding method, so it can be used more universally.

The final method is used in LAN types where the data is encoded using bit pairs, i.e. a 01 pair is used to represent a binary 0, and a 10 is used to represent a binary 1. The transition from high to low, or vice-versa is what determines the data type. This makes it easy to take advantage of this to set frame boundaries. Since all data is a transition of some type, the pairs 00 and/or 11 can be used to set frame boundaries, as they will never occur anywhere else. Obviously, this method can only be used on networks with the proper type of bit encoding. In general, a combination of methods are used to delimit frames, so as to lessen the chance of error.

In addition to the delimiters, all frames have source and destination address pairs. These are the hardware identifiers that name both the source and destination machines of the frame. The address pairs are placed at the beginning of the frame, so that they can be processed faster by a potential destination machine.

The final common part of frame structure is the data field. This is the area that carries the actual data for the frame. The length of this is dependent on the hardware protocol used, such as Ethernet, Token Ring, FDDI, ATM, etc.

Error Control

Since we are sending frames back and forth, we need to make sure that what we send is what we receive, which leads us to another job of the Data Link Layer, error control. There are a number of areas in which the Data Link Layer provides this service. The first is in the area of frame delivery. When a frame is transmitted, it is good to know that the frame arrived at all, and if so, that it arrived intact. There are two ways that the Data Link Layer handles this. When a frame is sent, and received successfully, the receiver sends back a special control frame whose purpose is to acknowledge the successful reception of the frame. This is commonly called an ACK frame, or just ACK. If the frame was not received successfully, then a negative acknowledgement, or NACK frame is sent. (It is useful to point out here that the ACK/NACK do not indicate the data in the frame is intact, but rather that the frame itself was received correctly. The Data Link Layer does provide error

detection for the data in the frame, we will cover that later. It is also good to note that the error correction capabilities of the Data Link Layer are optional in a protocol, and can be not used if this service is provided higher up.) This is useful when the frame is received, but what if the frame disappears completely? To handle this, timers are used. As each frame is sent, a timer is started. If the timer reaches its end before the ACK or NACK is received, then the frame is retransmitted. To avoid the receiver passing the same frame up to the Network Layer multiple times, a sequence number is assigned to each frame, so that the receiver can distinguish retransmissions from original frames.

In addition to the basic frame structure, the Data Link Layer can actually check to ensure that the physical bits in the frame at the receiving end are the exact same bits that were transmitted. There are two general ways to do this. The first is to send enough information along with the packet so that the receiver can figure out what the garbled character must have been. This procedure uses error-correcting codes. The second is to allow the receiver to deduce that an error occurred, but not what error or where, and request a retransmission of the frame. This method uses error-detection codes.

In error correction, the data is analyzed, and any errors are able to be fixed. The methods we will cover here are based upon the work of Hamming. In comparing two binary words, such as 10001001 and 10110001, it is relatively easy to determine which bits differ, by Exclusive ORing, or XOR'ing the words. (When two digits are compared via XOR, the result shows if the digits were alike or not. So 1 XOR 1 would give you a 0 as a result, since 1 and 1 are the same, whereas a 1 XOR 0 would give you a 1 as a result, since 1 and 0 are different.) In the case of our words, there are 3 bits that differ, so they are said to have a Hamming distance of 3. To help detect and fix errors, check bits are used. These are bits within the word that, instead of containing data, are used to protect the integrity of the data. This means that if you are sending eight — bit data words, that instead of their being 28 possible combinations of data, there will be fewer, as the check bits will take up space within that eight - bit word.

As an example, we take an eight bit word, and number each bit starting at the left with bit 1. Each bit that is a power of 2, (1,2,4,8), becomes a check bit, and the remaining bits are used to hold the data. The check bit is used to force the parity of a collection of bits, including itself to be even or odd. A given check bit can be used in multiple calculations. For example, the bit at position 7 is checked by bits 1, 2, and 4, ($1+2+4 = 7$). So when the word arrives at the receiver, the parity of each check bit is examined, (usually, 0 indicates even and 1 indicates odd.) If the parity of each check bit is correct, then the parity counter is not incremented. If the parity is incorrect, then the counter is incremented by the position of that bit. So, if bits 1,2,and 4 are incorrect, then the counter would read 7, and that would be the bit that is inverted. This allows the receiver to set bit 7 to its proper value. In general, Hamming codes can only detect single errors. However, by arranging the words as an array, and sending the words out

Mac Support Since 1985!

c-tree Plus[®]

ONE EMBEDDED DATABASE TOOL THAT FITS ALL YOUR APPLICATIONS

FairCom has been providing fast, flexible and scalable database development tools to the commercial developer for over 20 years. During this time FairCom has been utilized within countless embedded appliances, web server development projects and many vertical market applications. By offering industry leading performance, unsurpassed multi-user data availability and a complete transaction enabled database Server, FairCom provides the depth and breadth of technology to fit all of your database development needs. Add FairCom to your toolbox today.

ONE FAST ISAM TOOL, MANY PLATFORMS

Every copy of c-tree Plus supports all these platforms: Mac OS, Linux (Intel...), Novell Netware, OS/2, Solaris (SPARC), Solaris (Intel), Sun OS, AIX, HP UX, SCO, Interactive, AT&T Sys V, QNX, 88OPEN, FreeBSD, Windows 95/98/2000/NT, Lynx, Banyan Vines, and more...

Plus, support for
ADSP, SPX, TCP/IP



Embedded Hardware

APPLICATION	BENEFIT
Internet Appliances	Small footprint
Medical Devices	Stable
Factory Automation	Includes Full Source Code
Space Exploration	Proven Technology

...IN ONE
CONVENIENT
PACKAGE
FOR ONLY
\$895



FairCom
CORPORATION

Commercial Database Solutions Since 1979



Web Development

APPLICATION	BENEFIT
Dedicated Web Server	Robust Threading
B2B Server	Flexible Communication



Vertical Markets

APPLICATION	BENEFIT
Library Management	Fast
Insurance	Reliable
Inventory Control	Scalable
Point of Sale	Cross Platform

**FairCom
Offices**

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802

www.faircom.com • USA. 800.234.8180 • info@faircom.com

Other company and product names are registered trademarks or trademarks of their respective owners.
© 2000 FairCom Corporation

Scripter 2 with ScriptBASE

NEW!
Version 2.2

Tap the power of AppleScript with Main Event's Scripter!

**For professionals and novices
Webmasters and solution providers**

**No matter what your experience,
Scripter makes it easier!**

BEGINNER AT SCRIPTING?

- Unique command-builders help you learn to assemble commands
- Built-in tools for experimenting
- Shortcuts to speed scripting

EXPERIENCED WITH APPLESCRIPT?

- Unmatched single-step interactive debugging
- Watch and modify global and local variables while stepping
- Debug messages sent to script applications
- Includes ScriptBase to integrate scripting scenarios

*Scripter received MacWeek's 5-diamond rating – Twice!
"Scripter's virtuoso display of AppleScripting savvy and
debugging wizardry make it the best environment we've
found for learning or creating AppleScript scripts."
—MacWeek*

**Make AppleScript and your
applications work for you!**



Main Event
PO Box 21470
Washington, DC 20009
Tel: 202-298-9595
Sales: 800-616-8320
info@mainevent.com
www.mainevent.com

column-wise, instead of row-wise, the Hamming codes can then be used to correct any error in that array.

With error detection, the data is analyzed for correctness, but if an error is found, then the frame is retransmitted, rather than being fixed. One of the most common ways of doing this is via a Cyclic Redundancy Code, or CRC. In this case, the word is treated as a polynomial of k terms, with coefficients of 1 or 0 only. As an example, 11100011 would be an eight-term polynomial with coefficients of 1,1,1,0,0,0,1,1, being the equivalent of $x^7+x^6+x^5+x^1+x^0$. Within CRC, polynomial division is used to create the CRC code. The first part is to agree on a generator polynomial, $G(x)$ in advance. Both the high and low order bits of $G(x)$ must be 1. To compute the checksum of a frame with m bits, corresponding to a polynomial $M(x)$, the frame must be longer than $G(x)$. The idea here is to append a checksum onto the end of the frame in such a way that the polynomial represented by the checksummed frame is evenly divisible by $G(x)$. If there is a remainder after this division, the data is garbled somehow. The algorithm for this follows.

1) Let r be the degree of $G(x)$. Append r 0's to the end of the frame so that it contains $m + r$ bits, and corresponds to the polynomial $XrM(x)$.

Frame:	1101011011
$G(x)$:	10011
$XrM(x)$:	11010110110000

2) Divide $G(x)$ into $XrM(x)$ using Modulo 2 division

3) If there is a remainder, subtract that from $XrM(x)$ using Modulo 2 subtraction. The result is the checksummed frame to be transmitted, or $T(x)$. In our example, we would have a remainder of 1110. Subtracting 1110 from 11010110110000 via Modulo 2 subtraction, (which is simply subtraction via XOR) gives us a $T(x)$ of 11010110111110.

The ability of the CRC method to detect errors is extremely high. As an example, if 16-bit CRC is used, then all single or double bit errors are caught, all odd-numbered errors, all burst errors of length 16 or less, 99.997% of 17-bit error bursts, and 99.998% of all 18-bit or longer bursts.

Regardless of which method is used, both error correction and error detection both work well to help insure reliable data delivery across networks.

Media Access Control

The final job of the Data Link Layer is to handle access to the media itself. In most LANs, there are four basic methods for handling media access control:

- 1) Contention
- 2) Token Passing
- 3) Demand-priority
- 4) Switched

The first method, contention is the most widely used for now. This is the method used by such LAN types as Ethernet, Fast Ethernet, and 802.11 wireless networks. While widely used, this is also a fairly primitive form of media access control. Each time any device on a contention network needs to transmit data, it checks the wire to see if any other station is transmitting. If not, then the device can transmit, otherwise, the device must wait for the media to clear up. This type of network also requires that all devices transmit and receive on the same frequency band. The media can only support a single signal at a time, and this signal takes up the entire band. In other words, this is a baseband transmission network. This creates two important implications, the first being that only one device can transmit at a time, and that a device can transmit or receive, but not both at once. This is called a half-duplex operation.

In a contention based network, there are a number of things that are done to avoid collisions, which occur when two stations attempt to transmit at the same time. These are based on frame size and timing. As an example, with the IEEE 802.3 Ethernet, frame sizes are specified to be between 64 and 1524 octets in size. If a frame is going to be smaller than 64 octets, it is padded with 0s so that it is 64 octets in size. The reason for this ties into the timing part of collision avoidance, timing. If the minimum and maximum frame sizes are known, consistent quantities, then the amount of time it should take a frame to reach its destination can be accurately calculated. This time is the time it would take for each frame to propagate across the entire network. In a contention-based, baseband transmission network, each frame must be sent over the entire LAN to ensure that all recipients can receive it. In any case, the frame can be destroyed by a collision anywhere on the network. As the physical size, and number of devices on the network grow, the probability of collisions increase. One of the ways that modern networks avoid collisions is by using an inter-frame gap. This is a specified amount of dead air between frames. In a modern Ethernet network, the inter-frame gap is 96 bits long. So when a device transmits, not only must other devices wait for the frame to be transmitted, but also for the inter-frame gap to be transmitted. This gives the transmitting device time to either send another frame, or relinquish control of the media, without contending with all the other devices on the LAN.

Another method is through the Binary Exponential Backoff Algorithm. This is used after a collision occurs. After a collision, time is divided into discrete slots, which are this size of the time it takes a frame to travel round trip on the longest path allowed by the network type. In the case of 802.3 Ethernet, this is 51.2µsec. After the first collision, the stations that were affected wait either 0, or 1 of these time slots before trying again. If they collide again, they pick 0 - 3 time slots, wait, and try again. If a collision occurs again, then 0-7 time slots are used. In each case, this is a binary value, using 21 slots for the first collision, 22 for the second, 23 for the third and so on, hence the name of the algorithm. This occurs up to a maximum of 210 slots. Once that number of slots has been reached, then the stations wait a fixed amount of time. If 16 consecutive collisions occur at this point,

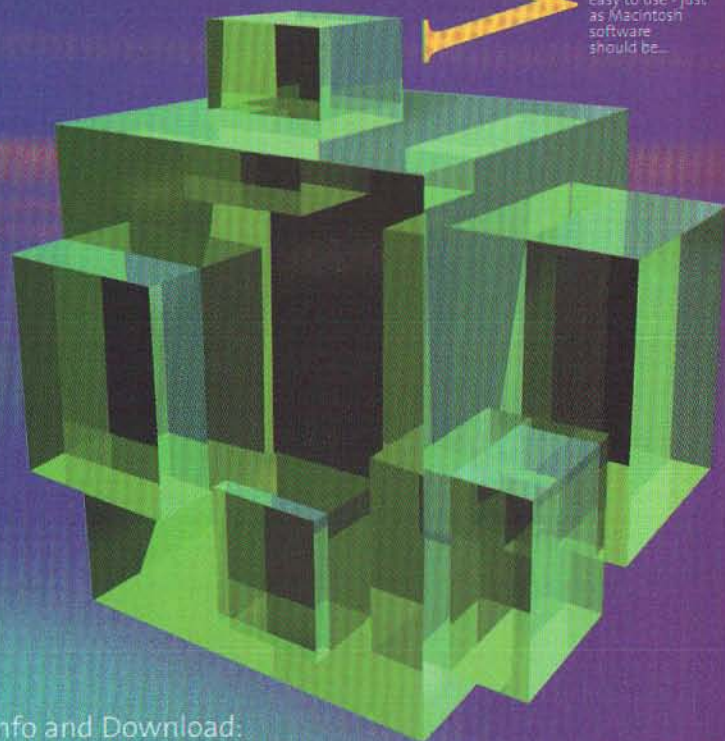
Version Control the Macintosh Way

VOODOO Server



The successor of our award-winning VOODOO. More powerful and still easy to use - just as Macintosh software should be.

The Version Control Tool for CodeWarrior Developers



Info and Download:
www.unisoftwareplus.com/products/voodoooserver.html



UNI SOFTWARE PLUS
 A-4232 Hagenberg, Austria
voodoo@unisoftwareplus.com
www.unisoftwareplus.com

the controller gives up, and reports a failure back to the computer. Further recovery is then up to higher layers.

Although collisions have a bad name, and rightfully so, they can be successfully managed through proper use of network design and devices, as well as using different types of network protocols in the areas where they are strongest.

The next type of media access control is via a token passing mechanism, used most often in Token Ring and FDDI networks. A token is a special frame that moves from device to device on the ring, (token - based networks use some sort of ring shape for their function.), and only circulates when the network is idle. The frame is only a few octets in length, and contains a special bit pattern. If a device needs to transmit data, it 'captures' the token, and converts that bit pattern into a Start Of Frame, (SOF) delimiter, that informs downstream devices that this is now a data - bearing frame, and that they have to wait until they get a token before they can transmit. The token is the sole way to access the network in this type of media access control. If a device receives the token, it has up to the default value in milliseconds for that network to convert the token into a data frame. If it does not have any data, it must release the token to the next device in the ring. If it does have data, then it converts the frame to a data frame, and begins transmitting. The recipient of the data then modifies the frame to show an ACK or NACK. Once the data transmittal is complete, the originating station converts the frame back to a token, and sends it back onto the network. The advantages to a token - passing network are highest in situations where a predictable delay in data transmittance is needed.

The obvious problem with a token-passing network is that of how to deal with the situation that arises when a transmitting station goes down, or drops off the ring. Without a method of dealing with this, the entire network could stop functioning, as there would be no way to convert the data frame back to a token frame. To handle this, the idea of a monitor station was provided. The monitor station monitors the ring, and ensures that the ring does not exceed a given time without the existence of a token. In the case we described above, the monitor, (which is usually the first station on the ring, or if that station goes down, a monitor contention algorithm decides the new monitor station.) grabs the data frame, removes it from the ring, or drains it, and issues a new token. By using monitor stations, we avoid having an orphan frame endlessly circulating, and preventing any other station from transmitting.

The third media access control method is that of Demand Priority Access Method, or DPAM. DPAM is a round - robin arbitration method wherein a central repeater, or hub, polls each port connected to it. This is done in port order, and identifies the ports with transmission requests. Once the ports that need to transmit are identified, then the priority of those ports is established. Normally, an idle port transmits an idle signal, indicating that it is not transmitting data. If a given port is cleared to transmit data, the hub tells it to cease transmission of the idle signal. When the port hears its own 'silence', it begins to transmit

data. Once data transmission begins, the hub alerts all ports connected to it that they may be receiving data. The hub then analyzes the destination address in the frame, compares it to its own internal link configuration table, and routes the frame to the port that connects to the specified device.

In a DPAM network, the priority is controlled by the central, or root hub. The overall priority for the network is also called the priority domain. This domain can include up to three levels of cascaded hubs. The central hub sends all traffic to the lower level hubs, which handle polling their own active ports once transmission has ceased. By using a priority mechanism, the problems of contention are avoided. No station can transmit twice in a row if other stations with equal, or higher priority requests are pending. If a station is transmitting, a station with a higher priority cannot interrupt that transmission. A higher priority request can however, preempt a lower level one. Finally, any lower level request that has waited for longer than 250ms is automatically raised to higher priority status. Although more reliable than contention networks, and cheaper than token - passing networks, DPAM was never a marketplace contender, and is virtually nonexistent in the modern LAN.

The final media access control method, switching, isn't as clearly defined as the other three, but is being used more and more in modern LANs to increase performance and efficiency. In essence, a switch decreases the network size for a given transmission to 3 devices: the switch, the transmitter, and the receiver. This increases performance by giving the data transmission the full bandwidth of the network for that transmission by creating a virtual network just for that transmission. It increases efficiency by decreasing the number of collisions on a contention network, and by allowing the use of the full bandwidth of the network. Switching has also allowed the use of the VLAN, or virtual LAN, where traffic can be segregated by protocol, port number, hardware address, etc. This also decreases the overall amount of traffic on the LAN, which decreases the error rate on the LAN. Switching can also be used by token-passing networks as well.

CONCLUSION

Well, we covered a lot for one layer, and really barely scratched the surface of Layer 2. There are whole books available that deal with this Layer alone, as it is one of the most complex layers in the OSI model, and one of the most critical. Hopefully, you now have a better idea of how this layer works, and why it is as important as it is. As usual, you are encouraged to read up on your own, the sources I list in my bibliography are a good start. Next time, Layer 3, the Network Layer!

BIBLIOGRAPHY AND REFERENCES

- Tannenbaum, Andrew S. Computer Networks. Third Edition Prentice Hall, 1996.
- Sportack, Mark. Networking Essentials Unleashed. SAMS Publishing, 1998.

**AS A
PROGRAMMER
OR SYSADMIN,
YOU'VE GOT
BETTER THINGS
TO DO THAN
FIGHT WITH A
WEB SITE**



It's time for you to take a look at MGI

MGI, a plug-in to 4D's award-winning server, WebSTAR, is designed to provide functionality to otherwise static web sites, whether for a private intranet or enterprise - class ISP/ASP. MGI was specifically developed to be used by web graphic designers with no scripting or programming experience. If you know HTML, you know MGI. And if you are a programmer, you can learn MGI by the time your pizza is delivered. You can try out MGI for free - right now - without obligation by downloading a fully - functioning demo at :

<http://www.pageplanetsoftware.com>

Searchable Databases
Online Stores
Info Baskets
Guestbooks
Banner Ads
Credit Card Transactions
Counters
Password Protection
Surveys
Quizzes
Form Processing
Conditionals
Math
Cookies

Date and Time
File Includes
File Writes
Web Based Email
Classified Ads
Discussion Groups
Web Chat
Affiliate Tracking
I/O Transactions
Δ Time Calculations
PGP Encryption
Token Tracking
Calendars
Random Rotation
Send Mail
E - Cards
Credit Bank



SOFTWARE BUILT WITH YOU IN MIND

by Tim Monroe

Somewhere I'll Find You

Working With Data References and Data Handlers

INTRODUCTION

In previous QuickTime Toolkit articles, we've learned that a movie is composed of tracks and that each track is associated with one specific kind of media data. During movie playback, QuickTime uses a media handler (a component of type `MediaHandlerType`) to interpret the media data and present it to the user in the appropriate manner. For example, if the media data for some particular track consists of compressed video data, the video media handler calls the decompressor specified in the image description and then draws the decompressed frames in the appropriate location. A media handler, however, does not typically concern itself with reading the media data from the movie file (or from wherever else the media data is stored). Instead, the media handler gets that data from a data handler (a component of type `DataHandlerType`), which is responsible for reading and writing a media's data. In other words, a data handler provides data input and output services for a media handler and for other parts of QuickTime.

We identify a source of media data to a data handler by providing a data reference. As we'll see shortly,

QuickTime currently includes four standard data handlers. Each data handler works with one specific sort of data reference. For instance, the URL data handler expects data references that are handles to URLs. That is to say, a URL data reference is a handle to a block of data that contains a NULL-terminated string of characters. The other data handlers expect their references in other forms.

A typical movie file contains data references to its media data in a data information atom, which is nested deep inside each track atom in the movie atom. In the previous article ("The Atomic Café" in MacTech, August 2000), we saw that a shortcut movie file contains a data reference atom that identifies an entire movie file. Data references can in fact pick out any kind of data, not just media data. In general, if we want to tell QuickTime where to find some data or where to put some data, we'll use a data reference to do so.

In this article, we're going to see how to work with each of the four standard data handlers. At the very simplest level, this involves nothing more than creating an appropriate data reference and putting that reference into a file (as we did last time) or passing it to a Movie Toolbox function like `NewMovieFromDataRef`. So we'll begin by learning how to create data references and call a few of the "FromDataRef" functions. Then we'll take a look at some of the functions that we can use to work with data handlers directly. These are fairly low-level functions that we won't need to use very often. Here, for fun, we'll see how to use them to write a general-purpose asynchronous file-transfer utility that relies solely on QuickTime APIs. Finally, toward the end of this article, we'll take a brief look at data reference extensions, which are blocks of additional data that we can associate with a data reference to assist QuickTime in working with the data picked out by the data reference. As we'll see, data reference extensions can be especially useful to graphics importers, to help them avoid having to take the time to validate some block of image data.

Tim Monroe's lizards have started laying eggs. Will they hatch? Will the parents eat the young? Stay tuned for further details. In the meantime, you can contact him at monroe@apple.com.

Our sample application this month is called QTDataRef; it illustrates how to work with data references and data handlers. **Figure 1** shows the Test menu from QTDataRef.

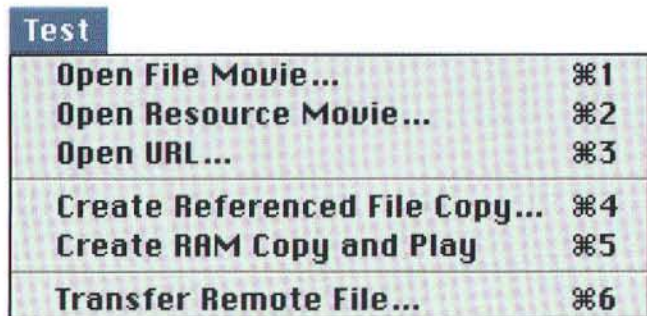


Figure 1. The Test menu of QTDataRef.

DATA HANDLER OVERVIEW

When QuickTime was first released, it included only one data handler, the file data handler, which can read and write data stored in files on a local file system. QuickTime version 2.0 introduced the handle data handler, which allows the Movie Toolbox to work with data stored in memory rather than in a file. QuickTime version 2.5 added the resource data handler, which can read data stored in a file's resource fork. More recently, QuickTime 3.0 added the URL data handler to support reading data from locations specified using uniform resource locators (URLs).

All data handlers are components of type `DataHandlerType`. Data handlers are distinguished from one another by their component subtypes. The file `Movies.h` defines constants for three of the four data handler subtypes:


```
enum {
    HandleDataHandlerSubType      = FOUR_CHAR_CODE('hdl'),
    ResourceDataHandlerSubType    = FOUR_CHAR_CODE('rsrc'),
    URLDataHandlerSubType         = FOUR_CHAR_CODE('url')
};
```

And the file `Aliases.h` defines the constant used for the file data handler:

```
enum {
    rAliasType                    = FOUR_CHAR_CODE('alis')
};
```

QuickTime also includes several other data handlers that it uses for its own private purposes and for which there is currently no public API. We won't consider these private data handlers in this article, but you should at least know that they exist (so that, for example, you aren't surprised if you iterate through all components of type `DataHandlerType` and find more than four of them).


A data reference is a handle to a block of memory that uniquely identifies the location of some media data for a QuickTime movie or some other data that QuickTime can manage. QuickTime currently provides support for four types of data references, one for each of the available data handlers. To let the cat out of the bag:



e-vue

The MPEG-4 Multimedia Technology and Service Company

Still Image and Streaming Media Technologies



MPEG-4 Multimedia Streaming - the Internet's Next Step

We invite you to join us at **e-Vue, Inc.** to create the MPEG-4 multimedia streaming products and services that are destined to revolutionize the Internet. MPEG-4 is a brand new ISO standard that, unlike all prior standards, was specifically developed to provide web-based multimedia content delivery and interactivity. **e-Vue, Inc.** is a pre-IPO Sarnoff Corporation spin-off chartered to commercialize Sarnoff's extensive software and patent portfolio of MPEG-4 technology. **e-Vue's** products and services provide the key enablers for high quality, reliable, and scalable delivery of video, audio, 2D and 3D graphics, over the Internet.

At **e-Vue** we are committed to career development for our employees and offer competitive salary, stock options and benefits packages. **e-Vue** provides employees with an extraordinary opportunity to grow in this fast-moving, high-tech industry. We are growing rapidly and looking for talented and motivated individuals to join our team.

- **Software Developers and Architects:** Windows, Mac, Unix
- **Expert Developers and Architects:** Networking, Video, Audio, Graphics

For full position details, please see our Website, <http://www.e-vue.com>.
To respond directly, submit resumes to:

- email: work@e-vue.com
- fax: (732) 452-9726
- mail: HR, e-Vue, Inc., 33 Wood Avenue South, 8th floor, Iselin, NJ 08830

e-Vue, Inc.

33 Wood Avenue South, 8th Floor
Iselin, NJ 08830
<http://www.e-vue.com>

- A file data reference is a handle to an alias record that specifies a file on a local storage volume (or on a remote storage volume mounted on the local machine). That is to say, a file data reference is an alias handle. Because the file data handler is of subtype `rAliasType` and uses alias handles as its data references, it is often called the alias data handler. In addition, because it originally handled files on the Macintosh hierarchical file system (HFS), it is also sometimes called the HFS data handler.
- A handle data reference is a handle to a 4-byte block of memory that holds a handle to some other block of data. That is to say, a handle data reference is a handle to a handle.
- A resource data reference is a handle to an alias record to which two pieces of information have been appended, a resource type and a resource ID. The target data is found in the resource fork of the file specified by that alias record, in the resource of the specified type and ID.
- A URL data reference is a handle to a NULL-terminated string of characters that comprise the URL. The string of characters should conform to the relevant IETF specifications; in particular, some non-alphanumeric characters may need to be encoded using the hexadecimal equivalents of their ASCII codes (for instance, the space character should be encoded as `"%20"`).

In the following four sections, we'll investigate these four types of data references in more detail. Before we begin, however, let's introduce a bit of terminology that will be useful throughout this article. Let's call the block of memory to which a data reference is a handle the referring data. And let's call the data picked out by the data reference the target data (or just the target) of the data reference. So, for example, the referring data of a URL data reference is the string of characters, and the target data of a URL data reference is the data in the file picked out by that URL.

THE FILE DATA HANDLER

We can use the file data handler to open movies that are specified using a file data reference, which is an alias handle. Listing 1 shows how to create a file data reference for a given file.

Listing 1: Creating a file data reference

```

Handle QTDR_MakeFileDataRef (FSSpecPtr theFile)
{
    Handle    myDataRef = NULL;

    QTNewAlias(theFile, (AliasHandle *)&myDataRef, true);

    return(myDataRef);
}

```

`QTDR_MakeFileDataRef` consists mainly of a call to the Movie Toolbox function `QTNewAlias`, which returns, in the location specified by the second parameter, an alias handle for the file specified by the first parameter. The third parameter is a Boolean value that indicates whether to create a minimal or a full alias record. (A minimal alias record contains only the minimum information needed to find a file; it's generally much faster to find the target of a minimal alias record, so that's what we'll use here.)

We could also have used the Alias Manager functions `NewAlias` (for a full alias record) or `NewAliasMinimal` (for a minimal alias record) to create the file data reference. The principal advantage of using `QTNewAlias` is that it returns an alias handle even if the file specified by the file doesn't yet exist. Both `NewAlias` and `NewAliasMinimal` return an error (`fnfErr`) — and do not create an alias record — if the specified file does not exist. As we'll see below, we sometimes want to create data references for files that we haven't yet created. (The Alias Manager does provide the `NewAliasMinimalFromFullPath` function that can create alias records for files that don't exist, but we'd rather not have to deal with full pathnames just to do that.)

Opening a Movie File

Let's consider a few examples of using file data references. First, we can pass a file data reference to the `NewMovieFromDataRef` function, to achieve exactly the same effect as calling the `OpenMovieFile` and `NewMovieFromFile` functions. Listing 2 defines the function `QTDR_GetMovieFromFile`, which creates a file data reference and then retrieves the movie in the specified movie file.

Listing 2: Opening a file specified by a file data reference

```

Movie QTDR_GetMovieFromFile (FSSpecPtr theFile)
{
    Movie    myMovie = NULL;
    Handle    myDataRef = NULL;

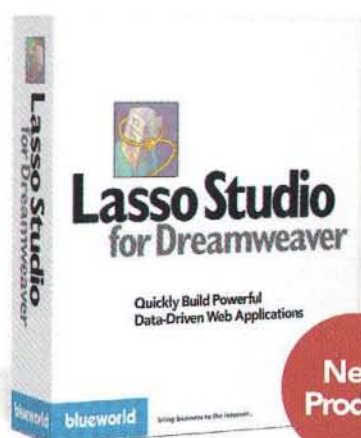
    myDataRef = QTDR_MakeFileDataRef(theFile);
    if (myDataRef != NULL) {
        NewMovieFromDataRef(&myMovie, newMovieActive, NULL,
                           myDataRef, rAliasType);
        DisposeHandle(myDataRef);
    }

    return(myMovie);
}

```

The last two parameters to the `NewMovieFromDataRef` function specify, respectively, the data reference and the data reference type. Since we are passing it a file data reference, we set the data reference type to `rAliasType`. Note that we call `DisposeHandle` to dispose of the data reference once we are finished using it. By the time `NewMovieFromDataRef` returns, the Movie Toolbox will have made a copy of the data reference, if necessary.

Quickly Build and Deploy Powerful Data-Driven Web Applications



New
Product



New 3.6
Version

Lasso Studio and Lasso Web Data Engine™ Lead The Way.

Building custom and shrink-wrapped database-driven Web applications requires a whole new way of doing things. Having pioneered the Web Data Engine™ over three years ago, Blue World and the Lasso Web Data Engine consistently lead the way providing a feature set Web developers describe as "incredible." Build online stores, discussion forums, resource management systems and other demanding database-driven Web applications with unrivaled performance, ease, security, extensibility, control and flexibility. Develop using multiple languages—including LDML, CDML, Server-Side JavaScript, Java and XML—and deploy across multiple platforms. Your Lasso code works identically regardless to which database you're connected. Lasso solutions for FileMaker® Pro databases easily scale to big iron ODBC-compliant databases like Oracle, Informix, Sybase and more with little or no change. What's more, the Lasso Java Application Programming Interface (LJAPI) provides developers an easy-to-use Java-based API for unprecedented extensibility.

Find out why hundreds of thousands of websites rely on award-winning Lasso technology for their business critical Web data. Download a 30-day evaluation copy at www.blueworld.com/download/ or order securely online today at the Blue World Store at store.blueworld.com.

Lasso Product Line – The leading Web tools for Macintosh and beyond.

bring business to the internet™

blueworld

Creating a Reference Movie File

Now let's turn to a more interesting example of using file data references. We learned in an earlier article ("Making Movies" in MacTech, June 2000) that a movie file might not contain the media data referenced by the movie's tracks; instead, that media data might be contained in some other file. A file that contains some media data is a media file, and the file that contains the movie atom is the movie file. (For simplicity, let's assume that our movies have only one track, a video track.) When the movie file and the media file are different files, the movie file is a reference movie file (because it refers to its media data and does not contain it). When the movie file and the media file are the same file, the movie file is a self-contained movie file. While generally we prefer to create self-contained movie files, it's useful to know how to create a reference movie file. (Reference movie files can be useful, for instance, if we want to share some media data among several QuickTime movies.)

One way to do this is to pass a file data reference to the `NewTrackMedia` function. Recall that `NewTrackMedia` is declared essentially like this:

```
Media NewTrackMedia ( Track theTrack,
                      OSType mediaType,
                      TimeScale timeScale,
                      Handle dataRef,
                      OSType dataRefType);
```

The fourth and fifth parameters specify the data reference and the data reference type of the file (or other container) that is to hold the media data for the specified track. Previously, whenever we called `NewTrackMedia`, we passed `NULL` and `0` in those parameters, to indicate that the media file should be the same as the movie file. Now we'll create a reference movie file simply by specifying a media file that is different from the movie file.

We're going to define a function `QTDR_CreateReferenceCopy` that we can use to create a copy of an existing movie that contains the first video track in the original movie. The copy's movie atom will be contained in one file (`theDstMovieFile`) and its media data will be contained in some other file (`theDstMediaFile`). `QTDR_CreateReferenceCopy` has this prototype:

```
OSErr QTDR_CreateReferenceCopy (Movie theSrcMovie,
                                FSSpecPtr theDstMovieFile, FSSpecPtr theDstMediaFile);
```

The definition of `QTDR_CreateReferenceCopy` will look vaguely familiar, at least if you recall the sequence of movie-making functions that we encountered in the `QTMM_CreateVideoMovie` function (in the "Making Movies" article). It uses `CreateMovieFile`, `NewMovieTrack`, `NewTrackMedia`, `BeginMediaEdits`, `EndMediaEdits`, and `AddMovieResource` in the standard ways. There are only two important differences between `QTDR_CreateReferenceCopy` and `QTMM_CreateVideoMovie`. First, we're going to obtain

the media data for the copy from the first video track in the existing movie; to do this, we'll call `GetMovieIndTrackType` and `GetTrackMedia` to get the media from the source movie:

```
mySrcTrack = GetMovieIndTrackType(theSrcMovie, 1,
                                   VideoMediaType, movieTrackMediaType);
mySrcMedia = GetTrackMedia(mySrcTrack);
```

Then we'll call `GetTrackDimensions` and `GetMediaHandlerDescription` to obtain some information about the source track and its media:

```
GetTrackDimensions(mySrcTrack, &myWidth, &myHeight);
GetMediaHandlerDescription(mySrcMedia, &myType, 0, 0);
```

When we create the copy movie, we'll need this information to be able to specify the size of the new track and the type of the track's media.

The second main difference between `QTDR_CreateReferenceCopy` and `QTMM_CreateVideoMovie` concerns the way we add media data to the new movie. Previously, we called an application-defined function to create the media samples and add them to the track's media (using `AddMediaSample`). Now, however, we've already got the media data for the new file — it's just the media data contained in the first video track of the source movie file. So, instead, we can use the Movie Toolbox function `InsertTrackSegment` to copy the media data from the source track to the new track, like this:

```
myErr = InsertTrackSegment(mySrcTrack, myDstTrack, 0,
                           GetTrackDuration(mySrcTrack), 0);
```

Here, `InsertTrackSegment` copies the entire source track (that is, starting at time 0 and extending for the duration `GetTrackDuration(mySrcTrack)`) and inserts it into the destination track starting at time 0. The result is that the new media file will contain a copy of the source track's media data.

There is one final detail we need to attend to. In order for the destination track to have the same visual characteristics as the source track, we also need to copy the source track matrix, clipping region, and graphics mode (among other things) into the destination track. Similarly, if the source track were a sound track, we'd need to copy its volume, sound balance, and other sound characteristics into the destination track. We could copy the visual settings by calling `GetTrackMatrix` and `SetTrackMatrix`, `GetTrackClipRgn` and `SetTrackClipRgn`, and so forth. Better yet, the Movie Toolbox provides the `CopyTrackSettings` function that copies all these settings in one fell swoop:

```
CopyTrackSettings(mySrcTrack, myDstTrack);
```

Listing 3 shows our complete definition of `QTDR_CreateReferenceCopy`.

Listing 3: Creating a reference movie file

```

QTDR_CreateReferenceCopy
OSErr QTDR_CreateReferenceCopy (Movie theSrcMovie,
    FSSpecPtr theDstMovieFile, FSSpecPtr theDstMediaFile)
{
    Track      mySrcTrack = NULL;
    Media      mySrcMedia = NULL;
    Movie      myDstMovie = NULL;
    Track      myDstTrack = NULL;
    Media      myDstMedia = NULL;
    Handle      myMediaRef = NULL;
    Fixed      myWidth, myHeight;
    OSType      myType;

    long      myFlags = createMovieFileDeleteCurFile |
        createMovieFileDontCreateResFile;
    short      myResRefNum = 0;
    short      myResID = movieInDataForkResID;
    OSErr      myErr = paramErr;

    // get the first video track and media in the source movie
    mySrcTrack = GetMovieIndTrackType(theSrcMovie, 1,
        VideoMediaType, movieTrackMediaType);
    if (mySrcTrack == NULL)
        goto bail;

    mySrcMedia = GetTrackMedia(mySrcTrack);
    if (mySrcMedia == NULL)
        goto bail;

    // get some information about the source track and media
    GetTrackDimensions(mySrcTrack, &myWidth, &myHeight);
    GetMediaHandlerDescription(mySrcMedia, &myType, 0, 0);

    // create a file data reference for the new media file
    myMediaRef = QTDR_MakeFileDataRef(theDstMediaFile);
    if (myMediaRef == NULL)
        goto bail;

    // create a file for the destination movie data and create an empty movie
    myErr = FSpCreate(theDstMovieFile, sigMoviePlayer,
        MovieFileType, 0);
    if (myErr != noErr)
        goto bail;

    // create a file for the destination movie atom
    myErr = CreateMovieFile(theDstMovieFile, sigMoviePlayer,
        smCurrentScript, myFlags,
        &myResRefNum, &myDstMovie);
    if (myErr != noErr)
        goto bail;

    // assign the default progress proc to the destination movie
    SetMovieProgressProc(myDstMovie, (MovieProgressUPP)-1, 0);

    // create the destination movie track and media
    myDstTrack = NewMovieTrack(myDstMovie, myWidth, myHeight,
        kNoVolume);
    myErr = GetMoviesError();
    if (myErr != noErr)
        goto bail;

    myDstMedia = NewTrackMedia(myDstTrack, myType,
        GetMediaTimeScale(mySrcMedia), myMediaRef, rAliasType);
    myErr = GetMoviesError();
    if (myErr != noErr)
        goto bail;

    // copy the entire source track into the destination track; this copies the track's media
    // samples into the destination media file

    myErr = BeginMediaEdits(myDstMedia);
    if (myErr != noErr)
        goto bail;

```

```

myErr = InsertTrackSegment(mySrcTrack, myDstTrack, 0,
    GetTrackDuration(mySrcTrack), 0);

    if (myErr != noErr)
        goto bail;

    CopyTrackSettings(mySrcTrack, myDstTrack);

    myErr = EndMediaEdits(myDstMedia);
    if (myErr != noErr)
        goto bail;

    // add the movie atom to the data fork of the movie file

    myErr = AddMovieResource(myDstMovie, myResRefNum,
        &myResID, NULL);

    bail:
    return(myErr);
}

```

We call the QTDR_MakeFileDataRef function to create a file data reference for the media file specified by the theDstMediaFile parameter and we pass that data reference to the NewTrackMedia function, as described earlier. By default, QTDataRef names the new movie file "untitled.mov" and the new media file "untitled.dat". When a user tries to open untitled.mov, the Movie Toolbox looks for untitled.dat to find the movie's media data. If it cannot find that file, the Movie Toolbox displays the dialog box shown in **Figure 2**.

Cross-Platform C++

PP2MFC puts your

PowerPlant™

applications on

Windows®

www.oofile.com.au

Database application generation from AppMaker, database compatible engine, Falcon e-tree Plus c++ interface with many added features; and a platform graphical, cross-platform report writer, XML, RTF, HTML output, PowerPlant interface; MFC interface, easy-to-use API, cross-platform file handling and directory iteration, source code, scales to client-server, single and multi-user royalty free, supports millions of records; server and standalone versions run on Mac, Windows & Unix; Database application generation from AppMaker, d-Base compatible engine, Falcon e-tree Plus c++ interface with many added features; cross-platform graphical, cross-platform report writer, XML, RTF, HTML output, PowerPlant interface; MFC interface, easy-to-use API; cross-platform file handling and directory iteration; source code, scales to client-server, single and multi-user royalty free; supports millions of records; server and standalone versions run on Mac, Windows & Unix; Database application generation from AppMaker, d-Base compatible engine, Falcon e-tree Plus c++ interface with many added features; cross-platform graphical, ...

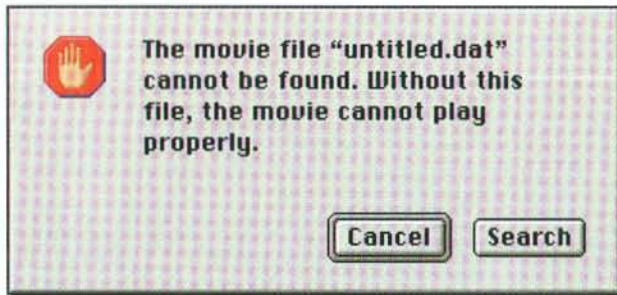


Figure 2. *The missing data dialog box.*

One reason we prefer to create self-contained movie files is to ensure that the media data is always available, so the user won't ever have to see this dialog box.

Before we move on, it's worth mentioning that the Movie Toolbox provides the `AddEmptyTrackToMovie` function, which we could use in Listing 3 instead of `NewMovieTrack`, `NewTrackMedia`, and `CopyTrackSettings`, like this:

```
myErr = AddEmptyTrackToMovie(mySrcTrack, myDstMovie,
                             myMediaRef, rAliasType, &myDstTrack);
if (myErr != noErr)
    goto bail;

myDstMedia = GetTrackMedia(myDstTrack);
```

`AddEmptyTrackToMovie` makes a copy of an existing track, either in the same movie or in a different movie; it doesn't actually copy the media data, but it does copy the track settings. The source code files accompanying this article provide an alternate version of the `QTDR_CreateReferenceCopy` function that uses `AddEmptyTrackToMovie`.

THE HANDLE DATA HANDLER

The handle data handler is used to read data from and write data to a location in memory. That location is specified using a Macintosh handle. A handle data reference is a handle to that handle. Listing 4 shows how to create a handle data reference.

Listing 4: Creating a handle data reference

```
Handle QTDR_MakeHandleDataRef (Handle theHandle)
{
    Handle    myDataRef = NULL;

    myDataRef = NewHandleClear(sizeof(Handle));
    if (myDataRef != NULL)
        BlockMove(&theHandle, *myDataRef, sizeof(Handle));

    return(myDataRef);
}
```

Here we allocate a relocatable block of memory that is the size of a handle (4 bytes) and then copy the handle passed to the function into that block of memory. The result is just what we want, a handle to the original handle. For an even simpler routine, we can replace the middle three lines of code by this one line:

```
PtrToHand(&theHandle, &myDataRef, sizeof(Handle));
```

The Memory Manager function `PtrToHand` allocates a new relocatable block of memory of the specified size and then copies the data specified by its first parameter into that block. In a little while, we'll encounter `PtrAndHand`, a cousin of `PtrToHand` that appends the data in a pointer to an existing handle.

The handle data handler is useful for many tasks. It's useful for playing movies that can fit entirely into RAM, and it's useful for handling images when the image data resides in memory and not in a file. In the latter case, we can just create a handle data reference from the handle that holds the image data and pass it to `GetGraphicsImporterForDataRef` to get a graphics importer than can manage the image data. Similarly, to play a movie stored in RAM, we can create a handle data reference and pass it to `NewMovieFromDataRef`. In this case, the block of memory referenced by the handle must contain the movie data and the movie atom.

To create a movie whose data is stored in RAM, we could use `NewTrackMedia` as illustrated in the previous section, passing it a handle data reference instead of a file data reference. Or, even more simply, we can exploit the ability of the `FlattenMovieData` function to flatten a movie into a location specified by a data reference instead of by a file specification record. The third parameter to `FlattenMovieData` is declared as a pointer to an `FSSpec` record, but we can set the flag `flattenFSSpecPtrIsDataRefRecordPtr` to instruct it to interpret that parameter as a pointer to a data reference record, defined by the `DataReferenceRecord` data type:

```
struct DataReferenceRecord {
    OSType          dataRefType;
    Handle          dataRef;
};
```

So all we need to do is create a handle data reference, fill in a data reference record with the appropriate information, and then pass the address of that record to `FlattenMovieData` in place of the `FSSpecPtr`. Listing 5 shows an excerpt from our `QTAApp_HandleMenu` function that handles the "Create RAM Copy and Play" menu item.

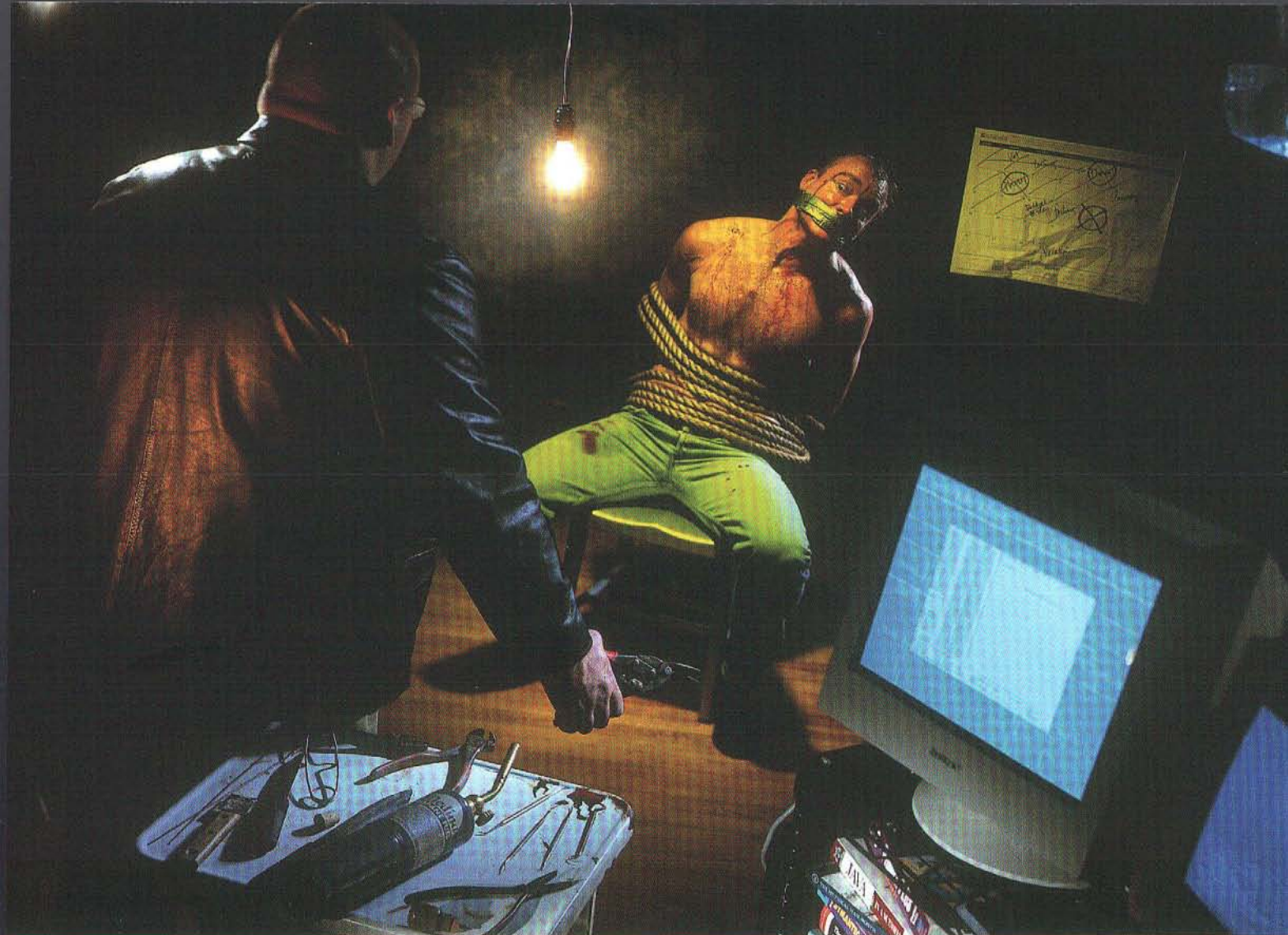
Listing 5: Creating a movie in RAM

```
case IDM_CREATE_RAM_COPY_AND_PLAY:
    if (myMovie != NULL) {
        Movie          myNewMovie = NULL;
        Handle         myDataRef = NULL;
        Handle         myHandle = NULL;
        DataReferenceRecord myDataRefRecord;

        myHandle = NewHandleClear(0);
        if (myHandle == NULL)
            goto bail;

        myDataRef = QTDR_MakeHandleDataRef(myHandle);
        if (myDataRef == NULL)
            goto bail;

        myDataRefRecord.dataRefType = HandleDataHandlerSubType;
        myDataRefRecord.dataRef = myDataRef;
```

Programming Doesn't Have To Be This Difficult. It's REALbasic!

NEW
Version!
REALbasic 2.1+
FREE
Update!

REALbasic is the award-winning, visual, object-oriented BASIC development environment for the Macintosh.

Use REALbasic's visual interface builder and platform-independent language to build native, compiled—not interpreted—professional quality applications in a fraction of the time it would take in C/C++. Because our language is platform-independent you only need to write a single set of code to create applications for both Macintosh and Windows. Leverage your C/C++ experience to extend REALbasic's capabilities using shared libraries, Mac OS Toolbox and Win32 API calls or by writing cross-platform REALbasic plug-ins. Create prototypes, Internet applications, database front-ends, even games with REALbasic! Its OOP language employs everything

you'd expect from a modern development environment—methods, properties, classes, subclassing, inheritance, constructors and destructors, virtual methods, and more. The IDE supports multi-threading, extensibility, TCP/IP controls, plus multimedia and QuickTime tools, and support for standards such as SQL, ODBC, AppleScript, and XCMDs. You can even import Visual Basic forms and modules.

Go to www.realbasic.com NOW to download a FREE trial version or call 512.263.1233.





Apple
Design
Award



2000 Runner-Up - Best Macintosh User Experience

*Free update for all owners of REALbasic 2.0 and above. REALbasic and the REALbasic logo are trademarks of REAL Software, Inc. Apple and the Apple logo are trademarks of Apple Computer, Inc., registered in the U.S., used with permission. All other trademarks are the property of their respective owners.


```

myNewMovie = FlattenMovieData( myMovie,
                                FlattenFSSpecPtrIsDataRefRecordPtr,
                                (FSSpecPtr)&myDataRefRecord,
                                sigMoviePlayer,
                                smSystemScript,
                                0L);
if (myNewMovie != NULL) {
    QTDR_PlayMovieFromRAM(myNewMovie);
    DisposeMovie(myNewMovie);
}

bail:
if (myHandle != NULL)
    DisposeHandle(myHandle);

if (myDataRef != NULL)
    DisposeHandle(myDataRef);
}

```

In this case, once we've created the movie in RAM, we call the function `QTDR_PlayMovieFromRAM` to play the movie in a window on the screen. Then we dispose of the new movie and the handle that contains the movie data, along with the handle data reference.

THE RESOURCE DATA HANDLER

The resource data handler was introduced in QuickTime version 2.5 to allow data to be read from a resource in a file's resource fork. Listing 6 shows how to create a resource data reference.

Listing 6: Creating a resource data reference

```

QTDR_MakeResourceDataRef
Handle QTDR_MakeResourceDataRef (FSSpecPtr theFile,
                                OSType theResType, SInt16 theResID)
{
    Handle    myDataRef = NULL;
    OSType    myResType;
    SInt16    myResID;
    OSErr     myErr = noErr;

    myDataRef = QTDR_MakeFileDataRef(theFile);
    if (myDataRef == NULL)
        goto bail;

    // append the resource type and ID to the data reference
    myResType = EndianU32_NtoB(theResType);
    myResID = EndianS16_NtoB(theResID);

    myErr = PtrAndHand(&myResType, myDataRef,
                      sizeof(myResType));
    if (myErr == noErr)
        myErr = PtrAndHand(&myResID, myDataRef, sizeof(myResID));

bail:
    if (myErr != noErr) {
        if (myDataRef != NULL)
            DisposeHandle(myDataRef);
        myDataRef = NULL;
    }

    return(myDataRef);
}

```

In `QTDR_MakeResourceDataRef`, we begin by creating a file data reference to the specified file. Then we call `PtrAndHand` to append the resource type to the referring data of that data reference, and then we call `PtrAndHand` once again to append the resource ID. Note that the resource

type and ID must be converted to big-endian format before being appended to the referring data.

We won't spend too much time considering the resource data handler, largely because we prefer not to keep our movie or image data in resource files (so that it's easily transportable to non-Macintosh operating systems). To show you that it works on Macintosh resource files, I've added the menu item "Open Resource Movie..." to the `QTDataRef` sample application; I've also included the file `ResBased.mov`, which contains a movie stored in a resource.

There is another reason why the resource data handler is of limited interest to us: it can read data from resource files but cannot write data to them. To see this, let's suppose that `myDataRef` is a resource data reference. Then we can call the `GetDataHandler` function like this:

```

myComponent = GetDataHandler(myDataRef,
                             ResourceDataHandlerSubType, kDataHCanWrite)

```

`GetDataHandler` returns the best data handler for the specified data reference and data handler type that provides the services specified by the third parameter. In this case, we're asking for a resource data handler that can write data. After this line of code completes, however, the value returned in `myComponent` will be `NULL`, indicating that there is no such resource data handler.

The last reason we're going to ignore the resource data handler in the future is perhaps the most obvious: if we do want to access some movie or image data stored in a resource, we can simply load that data into memory (by calling `GetResource`) and then use the handle data handler. So the resource data handler is largely redundant.

THE URL DATA HANDLER

We can use QuickTime to open movies and images that are specified using URLs. A URL is the address of some resource on the Internet or on a local disk. Typically, a URL is a string of characters like "http://www.apple.com". The initial portion of the URL, which precedes the first colon (:), is the URL's scheme or protocol. In this case, the scheme is "http", for the hypertext transfer protocol. QuickTime provides data handlers that can work with URLs whose scheme is "http", "ftp" (file transfer protocol), "rtsp" (real-time streaming protocol), or "file" (picking out a file on the local file system). Note that there really isn't just one URL data handler; there are in fact several data handlers that support URLs. For instance, the file data handler will ultimately be used to handle URLs whose scheme is "file". Still, we access these data handlers using URL data references.

As we've already seen, a data reference for the URL data handler is a handle to the NULL-terminated string of characters that comprise a URL. So it's relatively easy to create a URL data reference: just allocate a relocatable block of the appropriate size and copy the URL into that block.

Listing 7 defines the function QTDR_MakeURLDataRef, which creates a URL data reference for a given URL.

Listing 7: Creating a URL data reference

```

QTDR_MakeURLDataRef
Handle QTDR_MakeURLDataRef (char *theURL)
{
    Handle    myDataRef = NULL;
    Size      mySize = 0;

    // get the size of the URL, plus the terminating null byte

    mySize = (Size)strlen(theURL) + 1;
    if (mySize == 1)
        goto bail;

    // allocate a new handle and copy the URL into the handle

    myDataRef = NewHandleClear(mySize);
    if (myDataRef != NULL)
        BlockMove(theURL, *myDataRef, mySize);

bail:
    return(myDataRef);
}

```

If a URL picks out a movie file, we can call QTDR_MakeURLDataRef to create a URL data reference for it and then pass that data reference to the NewMovieFromDataRef function to open the specified movie file. Listing 8 defines the function QTURL_NewMovieFromURL that takes a URL and returns a movie identifier for the movie addressed by that URL. QTDR_GetMovieFromURL is just like QTDR_GetMovieFromFile, except that it uses URL data references.

Listing 8: Opening a movie specified by a URL

```

QTDR_GetMovieFromURL
Movie QTDR_GetMovieFromURL (char *theURL)
{
    Movie    myMovie = NULL;
    Handle    myDataRef = NULL;

    myDataRef = QTDR_MakeURLDataRef(theURL);
    if (myDataRef != NULL) {
        NewMovieFromDataRef(&myMovie, newMovieActive, NULL,
                           myDataRef, URLDataHandlerSubType);

        DisposeHandle(myDataRef);
    }

    return(myMovie);
}

```

The QTDR_GetMovieFromURL function is a large part of what we need to handle the "Open URL..." menu item (supported both by QuickTime Player and by our sample application QTDataRef). But it isn't quite all of what we need. First, of course, when the user selects "Open URL...", we need to obtain a URL from him or her. QuickTime Player displays the dialog box shown in **Figure 3**, which contains space to type in a URL, as well as a pop-up menu containing a list of recently opened URLs.

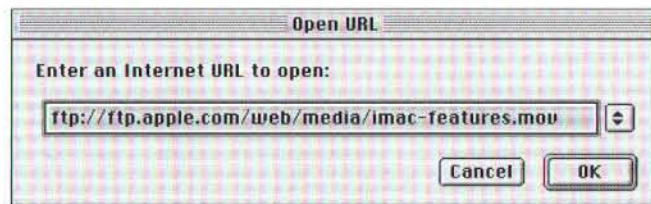


Figure 3. The Open URL dialog box of QuickTime Player.

For the moment, we'll be content to display the dialog box shown in **Figure 4**, which does not provide the pop-up menu. (Providing a pop-up menu or some other kind of list for recently accessed URLs is a good idea, however, since URLs are often longer than 255 characters, which is the most our current code will support.)

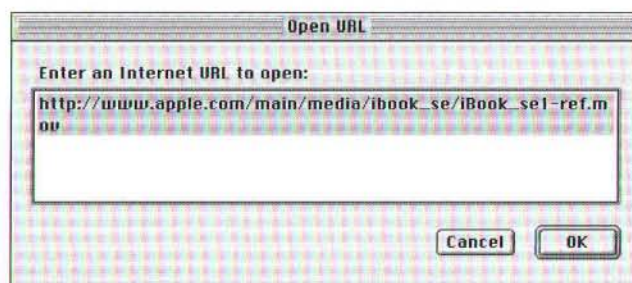


Figure 4. The Open URL dialog box of QTDataRef.

GOT BUGS?
...and you're drowning in paper
trying to keep track of them?

You need BugLink!

- **BugLink is a client/server application** allowing you to connect developers, testers, and support engineers anywhere in the world.
- **Intuitive user interface** gives you the information you need at a glance.
- **Fully customizable database**—add the fields you need to each project.
- **Custom TCP/IP protocol minimizes network traffic**—ideal for dial-up connections.
- **Client applications can operate 'off-line'**—allowing bug entry and modification even when not connected to the network!
- **Cross platform**—set up mixed Macintosh and Windows environments in minutes with no additional software needed!
- **Try before you buy.** Download and try risk free for 30 days from <http://www.pandawave.com/bl/>

A 5 User License starts at \$299; that's only \$60 per person.

The PandaWave
<http://www.pandawave.com>

The code to display and manage this dialog box is contained in the function `QTDR_GetURLFromUser`, which we won't consider in detail here; it's really quite similar to the function `QTInfo_EditAnnotation`, which we considered in a previous article (see "The Informant" in MacTech, July 2000). It's also useful to get the basename of the URL so that our movie window has a title (the basename is the portion of the URL following the rightmost URL separator). See the file `QTDataRef.c` for the function `QTDR_GetURLBasename`, which we use to find that basename.

FILE TRANSFER

We've learned that QuickTime supplies a data handler that can read data from remote locations specified by a URL and a data handler that can write data into files on the local file system. We can use both of these data handlers at the same time to read data stored remotely on the Internet and write it into a local file. This gives us, in effect, a network file-transfer capability that operates using only QuickTime APIs. Moreover, QuickTime supports calling these data handlers asynchronously, so that our application can perform other processing while the file transfer is taking place. In this section we'll see how to do all this.

Our ulterior motive here is to get a glimpse of the lower-level functions supported by data handlers. Hitherto, we've pretty much always used data handlers indirectly, by passing data references to Movie Toolbox or ICM functions, or simply by embedding data references in files. Now we want to see how to work with data handlers directly.

In overview, our file transfer will proceed like this: get an instance of the URL data handler and configure it to copy data from a remote file into a memory buffer. Then get an instance of the file data handler and configure it to copy data from that buffer into a local file. Then keep copying data into and out of the buffer until the entire remote file has been copied. To make it easy for this to work asynchronously, our file-transfer code is going to rely on a few global variables, declared like this:

```
Ptr          gDataBuffer = NULL;
ComponentInstance gDataReader = NULL;
ComponentInstance gDataWriter = NULL;
DataHCompletionUPP gReadDataHCompletionUPP = NULL;
DataHCompletionUPP gWriteDataHCompletionUPP = NULL;
long          gBytesToTransfer = 0L;
long          gBytesTransferred = 0L;
Boolean       gDoneTransferring = false;
```

The variables `gDataReader` and `gDataWriter` are the component instances of the URL data handler that reads data and the file data handler that writes data. `gDataReader` puts data into the buffer `gDataBuffer`, whence `gDataWriter` gets the data that it writes into the local file. The asynchronous nature of the file transfer is achieved largely by two completion functions, to which `gReadDataHCompletionUPP` and `gWriteDataHCompletionUPP` are universal procedure pointers. Finally, the global variables `gBytesToTransfer`,

`gBytesTransferred`, and `gDoneTransferring` keep track of information while the transfer is underway.

Creating the Local File

The first thing we need to do, of course, is create the local file into which the remote file data is to be copied. Let's suppose that `theFile` is a file system specification for the local file (which we perhaps got from the user by calling our framework function `QTFrame_PutFile`). We should first delete that file, if it already exists, by calling `FSpDelete`:

```
FSpDelete(theFile);
```

If the specified file doesn't exist, `FSpDelete` will return the error code `fnfErr`, which we can safely ignore. Then we can call `FSpCreate` to create a new empty file, like this:

```
myErr = FSpCreate(theFile, kTransFileCreator, kTransFileType,
                  smSystemScript);
```

(Note that we are hard-coding the file's type and creator codes here; I'll leave it as an exercise for you to concoct a more intelligent way to set these values.)

Opening and Configuring Data Handlers

Before we can work with a data handler, we need to open an instance of the appropriate data handler component. We saw earlier that we can find a particular data handler by calling `GetDataHandler`, passing it a data reference, a type, and a set of flags that indicate the data-handling services we need it to provide. If `theURL` is a C string that specifies the remote file and `theFile` is a pointer to a file system specification for the local file into which the remote file data is to be copied, we can create the required data references like this:

```
Handle myReaderRef = QTDR_MakeURLDataRef(theURL);
Handle myWriterRef = QTDR_MakeFileDataRef(theFile);
```

And we can open the appropriate data handlers with this code:

```
gDataReader = OpenComponent(GetDataHandler(myReaderRef,
                                             URLDataHandlerSubType, kDataHCanRead));
gDataWriter = OpenComponent(GetDataHandler(myWriterRef,
                                             rAliasType, kDataHCanWrite));
```

As you can see, we're asking for a URL data handler that can read data and a file data handler that can write data. Once we've got our data handler instances, we need to configure them by telling them which data references they are going to work with. We also need to have them open a connection to the targets of those data references. We can set the data references by calling `DataHSetDataRef`, like this:

```
myErr = DataHSetDataRef(gDataReader, myReaderRef);
myErr = DataHSetDataRef(gDataWriter, myWriterRef);
```

Then we can open the appropriate connections by calling `DataHOpenForRead` and `DataHOpenForWrite`:

```
myErr = DataHOpenForRead(gDataReader);
myErr = DataHOpenForWrite(gDataWriter);
```


*"Never forget that the most powerful
force on earth is love."*

—Nelson A. Rockefeller

Move over love.



Get a new Apple PowerMac™ G4
or one of our refurbished models
at price you will L-O-V-E.



**Small Dog
Electronics**

1673 Main Street
Waitsfield, VT 05673 USA
Phone: **802-496-7171**
Online: **smalldog.com**



Apple Specialist

Transferring Data Synchronously

It might seem like we haven't done much work yet, but (believe it or not) we are ready to begin transferring data from the remote file into the local file. At this point, we have a choice: we can transfer the data synchronously (that is, waiting for all the data to arrive before continuing with any other work) or asynchronously. Ultimately we want to have our transfers proceed asynchronously, but let's take a moment to see how to do them synchronously.

To perform a synchronous transfer, we can call `DataHGetData` and `DataHPutData`. These functions require the intermediate buffer to be accessed by a handle (not by a pointer, as with asynchronous transfers). We can call `DataHGetFileSize` to see how many bytes we need to transfer and then allocate a handle of that size:

```
myErr = DataHGetFileSize(gDataReader, &gBytesToTransfer);
Handle gDataBuffer = NewHandleClear(gBytesToTransfer);
```

If `gDataBuffer` is successfully created, we can then transfer the remote file to the local file with these four lines of code:

```
DataHGetData(gDataReader, gDataBuffer, 0L, 0L,
             gBytesToTransfer);
DataHCloseForRead(gDataReader);
DataHPutData(gDataWriter, gDataBuffer, 0L, NULL,
             gBytesToTransfer);
DataHCloseForWrite(gDataWriter);
```

All we need to do is fetch the remote data into the intermediate buffer, close the connection to the remote file, write the data into the local file, and then close the connection to the local file. This code assumes, of course, that we can allocate a buffer large enough to hold the entire remote file. With very little work, you could generalize this code to work with files too large to fit entirely into the available RAM. (We'll do this below, when transferring asynchronously.)

So there you have it: with barely a dozen lines of code, we've managed to copy a file located somewhere out on the Internet into a local file. The downside here is that the transfer occurs synchronously and is limited to files that can fit entirely into RAM. Let's remove these two limitations.

Transferring Data Asynchronously

We can remove the limitation on file size rather simply, by allocating an intermediate buffer of a set size and then reading and writing chunks of data of that size. We'll allocate a buffer that is 10 kilobytes, like this:

```
#define kDataBufferSize 1024*10
gDataBuffer = NewPtrClear(kDataBufferSize);
```

So all we need to do is read a chunk of data of size `kDataBufferSize`, write that chunk of data, read another chunk of data, write that chunk, and so on until the entire remote file is transferred. Of course, eventually we'll probably end up with a chunk that's smaller than `kDataBufferSize`,

so we'll need to keep track of how much of the file remains to be transferred and adjust our reads accordingly.

We can achieve an asynchronous transfer by using the functions `DataHReadAsync` and `DataHWrite` to read and write data. These functions queue up a read or write request to the data handler and then return immediately, without waiting for the request to be serviced. The request will be serviced at some later time, when we call `DataHTask` to task the data handler. Once the request is serviced, the data handler executes a data handler completion function that we specify when we call `DataHReadAsync` or `DataHWrite`.

A data handler completion function takes three parameters: a pointer to the buffer into which data was written or from which data was read, an application-specific reference constant, and an error code. We'll ignore the error code and use the reference constant to hold the number of bytes just written or read by the data handler.

We begin the file transfer by reading some data from the remote file. We could simply call `DataHReadAsync` directly, passing it the appropriate parameters. Instead, however, we'll be a bit clever here and call our write completion function `QTDR_WriteDataCompletionProc`, passing it parameters that indicate that we've just successfully finished writing 0 bytes:

```
QTDR_WriteDataCompletionProc(gDataBuffer, 0L, noErr);
```

The reason for this is simple: our `QTDR_WriteDataCompletionProc` function (defined in Listing 9) contains code for figuring out how many bytes to request and then for issuing that request.

Listing 9: Responding to a write operation

```
QTDR_WriteDataCompletionProc
PASCAL_RTN void QTDR_WriteDataCompletionProc
                (Ptr theRequest, long theRefCon, OSErr theErr)
{
    #pragma unused(theErr)
    long   myNumBytesToRead;
    wide   myWide;

    // increment our tally of the number of bytes written so far
    gBytesTransferred += theRefCon;

    if (gBytesTransferred < gBytesToTransfer) {
        // there is still data to read and write, so schedule a read operation

        // determine how big a chunk to read
        if (gBytesToTransfer - gBytesTransferred >
            kDataBufferSize)
            myNumBytesToRead = kDataBufferSize;
        else
            myNumBytesToRead = gBytesToTransfer -
                               gBytesTransferred;

        myWide.lo = gBytesTransferred; // read from the current offset
        myWide.hi = 0;

        // schedule a read operation
        DataHReadAsync(gDataReader,
                       theRequest, // the data buffer
                       myNumBytesToRead,
                       &myWide,
                       gReadDataHCompletionUPP,
                       myNumBytesToRead);
    }
}
```



```

} else {
// we've transferred all the data; set a flag to tell us to close down the data handlers
gDoneTransferring = true;
}
}

```

As you can see, we first update the global variable `gBytesTransferred` that keeps track of the number of bytes already transferred. Then we figure out how many bytes remain to be read from the remote file; we read that number of bytes, if it's less than the size of our intermediate buffer, or else we read an entire buffer of data. Finally, we call `DataHReadAsync` to schedule a read operation. Notice that we specify the `gReadDataHCompletionUPP` as the read completion function.

Once some data is read into the local buffer, our read completion function will be executed. Listing 10 shows our read completion function. It's even simpler than the write completion function; all it does is schedule a write operation to copy the data from the buffer into the local file.

Listing 10: Responding to a read operation

```

QTDR_ReadDataCompletionProc
PASCAL_RTN void QTDR_ReadDataCompletionProc
(Ptr theRequest, long theRefCon, OSErr theErr)
{
#pragma unused(theErr)
// we just finished reading some data, so schedule a write operation
DataHWrite(gDataWriter,
theRequest,           // the data buffer
gBytesTransferred,    // write from the current offset
theRefCon,           // the number of bytes to write
gWriteDataHCompletionUPP,
theRefCon);
}

```

In this case, `theRefCon` contains the number of bytes just read from the remote file, which is the number of bytes that should be written to the local file. Notice that now we specify `gWriteDataHCompletionUPP` as the write completion function. The read and write completion functions keep scheduling write and read requests, specifying each other as the completion function for those requests. So we keep successively reading and writing data, until the entire file is transferred.

Tasking the Data Handlers

There is one final step needed to make this all work. Namely, we need to give the data handlers some processor time to do their work. We do this by periodically calling `DataHTask`. On the Macintosh, we can insert calls to `DataHTask` into the application function `QTAApp_HandleEvent`, which is called by our application framework every trip through the main event loop. Listing 11 shows the definition of `QTAApp_HandleEvent` in `QTDataRef`.

Listing 11: Tasking the data handlers

```

QTAApp_HandleEvent
Boolean QTAApp_HandleEvent (EventRecord *theEvent)
{
#pragma unused(theEvent)

// if we're done, close down the data handlers
if (gDoneTransferring)
QTDR_CloseDownHandlers();
}

```

```

// give the data handlers some time, if they are still active
if (gDataReader != NULL)
DataHTask(gDataReader);

if (gDataWriter != NULL)
DataHTask(gDataWriter);

return(false);
}

```

If the file is done being transferred, then `QTAApp_HandleEvent` calls the function `QTDR_CloseDownHandlers` (defined later) to close things down. Otherwise, it calls `DataHTask` on both the URL data handler and the file data handler. At most one of them will have some work to do, but it doesn't hurt to task both of them.

Our Windows framework does not however call `QTAApp_HandleEvent` periodically, so we'll have to do that ourselves. Probably the easiest way is to install a timer task, like this:

```

gTimerID = SetTimer(NULL, 0, kQTDR_TimeOut,
(TIMERPROC)QTDR_TimerProc);

```

The timer callback function `QTDR_TimerProc`, defined in Listing 12, simply calls `QTAApp_HandleEvent`.

Listing 12: Handling timer callbacks

```

QTDR_TimerProc
void CALLBACK QTDR_TimerProc (HWND theWnd, UINT theMessage,
UINT_PTR theID, DWORD theTime)
{
#pragma unused(theWnd, theMessage, theID, theTime)
QTAApp_HandleEvent(NULL);
}

```

StoneTable

You thought it was **just** a replacement
for the List Manager ?

We lied, it is **much** more !

Tired of always adding just one more feature to your LDEF or
table code ? What do you need in your table ?

Pictures and Icons and Checkboxes ?
adjustable columns or rows ?
Titles for columns or rows ?
In-line editing of cell text ?
More than 32K of data ?
Color and styles ?
Sorting ?
More ??

How much longer does the list need to be to make it worth
\$200 of your time ?

See just how long the list is for StoneTable.

Make StoneTable part of your toolbox today !

Only \$200.00

MasterCard & Visa accepted.

StoneTablet Publishing

More Info & demo

Voice/FAX (503) 287-3424

<http://www.teleport.com/~stack>

stack@teleport.com

The products you need, with the prices and service you deserve... guaranteed.

Power Tools for Programmers!

CodeWarrior Pro 6

NEW!

CodeWarrior for Mac OS is a powerful integrated development environment that includes a fully object-oriented application development framework called PowerPlant. With CodeWarrior for Mac OS you can quickly develop reliable, professional quality applications that execute on Classic Mac OS, or OS X.

NEW Version — In Stock Now!

AS LOW AS

\$359

Spotlight

Spotlight is the first Macintosh "Automatic Debugger". It can automatically locate run time errors in your code and display the offending source code line. Unlike similar tools on other platforms Spotlight is easy to use. No source code changes are necessary for application debugging. Spotlight can automatically check for wild pointers, memory leaks, overwrites, underwrites, invalid dereferencing of handles, and even toolbox parameter validity checking — spotlight knows Macintosh verifying parameters to over 400 toolbox calls.



\$189

Resorcerer 2.2

Resorcerer is the only supported general-purpose resource editor for Macintosh. Relied upon by thousands of Mac developers, Resorcerer features a wealth of powerful yet easy-to-use tools for easier, faster, and safer editing of Macintosh data files and resources. Whether you have to parse a picture, debug a data fork, design and try out Balloon Help, create a scripting dictionary, create anti-aliased icons, design and edit a custom resource with 40,000 fields in it, create C source code to run a dialog, or any of hundreds of other resource-related tasks, Resorcerer's magic will quickly save you time and money.



\$256

REALbasic 2.1

NEW!

REALbasic is the programming tool for "the rest of us." Each window type, control, and menu is preconfigured and instantly works as it should. The drag and drop Window Editor allows you to quickly and easily create your application's interface so you can focus on the important part—your creativity. Includes 900 page manual, examples and tutorial on CD ROM! The professional version has all the power of the standard, plus database support and allows you to cross compile your code for Windows with a single click!

AS LOW AS

\$139

VOODOO Server

VOODOO Server is a version control system for software developers using Metrowerks CodeWarrior under Mac OS. VODOO Server and the corresponding VODOO clients (the included CodeWarrior VCS plug-in and the VODOO Admin application) are designed to offer reliable and robust version control features while minimizing the administrative overhead that usually accompanies version control. If you're a single programmer or managing a team of developers, version control can make or break your project. Do it right, with VODOO.



\$79

Future BASIC 3

NEW!

One of the most flexible and powerful development environments on the Macintosh today! Easily create programs with the visual program editor, drop into the BASIC editor to define powerful logic with the worlds easiest programming language, or work directly with the Macintosh toolbox. The only BASIC compiler on the market that gives you 100% access to all of the power of the Macintosh toolbox!



\$159

and hundreds more!

PageCharmer 2.0
\$139

Scripter 2.0
\$179

WebTen
\$349

ToolsPlus Lite
\$99

FaceSpan 3.0
\$179

WebSpice 1,000,000
\$89

WebSpice Animations
\$89

MkLinux
\$39

PowerKey Rebound
\$89

Developer DEPOT®

PO Box 5200 • Westlake Village, CA • 91359-5200 • Voice: 800/MACDEV-1 (800/622-3381)
Outside US/Canada: 805/494-9797 • Fax: 805/494-9798 • E-mail: orders@devdepot.com

www.devdepot.com

Master the Web!

WebSTAR Server Suite 4.3

WebSTAR Server Suite is a complete set of powerful and easy-to-use Internet servers for the Mac OS. Effortlessly serve web pages, host email accounts, publish databases, and share files - all with a single application on one Mac! WebSTAR Server Suite is perfect for Internet or Intranet serving, single or multiple sites, small and large businesses - it's power and ease-of-use saves any organization time and money.



\$539

CyberGauge 3.0

Monitor the bandwidth usage of up to five different machines on your network! Do you need to upgrade your webserver? How hard is your eMail server working? Are you getting all the bandwidth you're paying for? Not only can CyberGauge answer all these questions, new features allow CyberGauge to eMail or page your network device becomes unresponsive or passes a threshold of usage you define - an essential first line of defense for early detection of denial of service attacks and necessity for warning you and tracking quality of ISPs that may have brown outs and shutdowns.



\$249

Funnel Web Pro

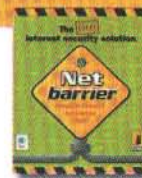
Funnel Web is the ultimate web analysis solution for professionals. Specifically designed for profiling web site usage and monitoring customer usage patterns, Funnel Web is ideal for examining server performance and online effectiveness. Funnel Web can analyze log file formats from any server, WebSTAR, WebTen, even Unix or NT hosted servers. Discover the most popular pages on your site, track server loads & optimize server performance, profile visitors based on organization, domain name, country, browser, etc. Funnel Web does it all!



\$299

NetBarrier

NetBarrier offers a Personal Firewall, Antivandal protection, and Internet Filtering. Protect your machine from intrusions by Internet or AppleTalk. Incorrect passwords and individual actions are logged, you are alerted to hostile actions, and intruders are easily isolated. Internet Filtering allows you to be sure that passwords, credit card numbers, and other sensitive information can never be exported from your computer - the content itself is filtered before any transfer! (Rated 4 mice by Macworld Magazine)



\$57⁹⁵

...and great hardware solutions!

2 USB PCI Card

Add two USB ports to your older Macintosh. Connect up to 127 devices to the Universal Serial Bus (USB) that is Apple's new standard for desktop connectivity. USB mouse devices, keyboards, joysticks, game controllers, printers, scanners - connect them all to your current computer. Installs in minutes!



\$32⁹⁵

Dr. Bott Moni Switch ADB or USB

Do you need 4 monitors and 4 keyboards for your 4 servers? With Dr. Bott Moni-Switch you can connect a single keyboard and monitor to up to 4 machines at once! A simple flick of a switch directs the video input and keyboard commands to the appropriate CPU! Available in USB and ADB models, with 2 or 4 machine support, and bundles with USB PCI cards so you can mix and match USB and ADB machines with the same Moni-Switch! Great for programmers to do back ground compiles, ideal for server rooms overcrowded with monitors and keyboards!



As low as
\$129⁹⁵

Macsense USB Full Size Keyboard

Just plug this keyboard into your Mac and start typing! The UKB-600 keyboard from Macsense is designed to get you typing quickly and easily, without any hassle or compatibility worries. It features two tone translucent design, colored to match your favorite flavor of Macintosh. It offers soft touch with positive tactile feedback and build built in USB port on either side of the keyboard. Includes a 5' USB cable and is 100% Macintosh compatible, simply plug and play, as easy as Macintosh!



\$44⁹⁵

Macsense Internet Sharing Router

Looking to get your whole office online without shelling out thousands of dollars? If so, the XRouter Internet Sharing Hub offers the perfect solution. This amazing Ethernet-to-Ethernet hub connects an entire network of up to 252 users to the Internet using only one ISP account and one Cable or DSL modem!



\$199

We remove the timer task once the data transfer is completed (see Listing 13 below).

It's important to know that a data handler may execute our requests to read and write data asynchronously or not, even if we specify completion functions. If the handler decides to operate synchronously, then it will not return immediately when we call `DataHReadAsync` or `DataHWrite`; instead, it will perform the requested operation and then return. We still need to call `DataHTask`, however, to give the data handlers an opportunity to execute their completion functions.

Finishing Up

When the write completion function `QTDR_WriteDataCompletionProc` determines that all the data has been read from the remote file and written into the local file, it sets the global variable `gDoneTransferring` to true. When `QTAApp_HandleEvent` is called and `gDoneTransferring` is true, `QTAApp_HandleEvent` calls `QTDR_CloseDownHandlers`, defined in Listing 13.

Listing 13: Closing down the data handlers

```
QTDR_CloseDownHandlers
void QTDR_CloseDownHandlers (void)
{
    if (gDataReader != NULL) {
        DataHCloseForRead(gDataReader);
        CloseComponent(gDataReader);
        gDataReader = NULL;
    }

    if (gDataWriter != NULL) {
        DataHCloseForWrite(gDataWriter);
        CloseComponent(gDataWriter);
        gDataWriter = NULL;
    }

    // dispose of the data buffer
    if (gDataBuffer != NULL)
        DisposePtr(gDataBuffer);
    // dispose of the routine descriptors
    if (gReadDataHCompletionUPP != NULL)
        DisposeDataHCompletionUPP(gReadDataHCompletionUPP);
    if (gWriteDataHCompletionUPP != NULL)
        DisposeDataHCompletionUPP(gWriteDataHCompletionUPP);
    gDoneTransferring = false;

#ifdef TARGET_OS_WIN32
    // kill the timer that tasks the data handlers
    KillTimer(NULL, gTimerID);
#endif
}
```

`QTDR_CloseDownHandlers` simply closes the connections to the local and remote files and then closes the component instances of the data handlers. On Windows, it also removes the timer task that was calling `QTAApp_HandleEvent` periodically.

Listing 14 contains the complete definition of the `QTDR_CopyRemoteFileToLocalFile` function, which is called in response to the "Transfer Remote File..." menu item.

Listing 14: Copying a remote file into a local file

```
QTDR_CopyRemoteFileToLocalFile
OSErr QTDR_CopyRemoteFileToLocalFile
(char *theURL, FSSpecPtr theFile)
```

```
Handle          myReaderRef = NULL; // data ref for the remote file
Handle          myWriterRef = NULL; // data ref for the local file
ComponentResult myErr = badComponentType;

// delete the target local file, if it already exists;
// if it doesn't exist yet, we'll get an error (InflErr), which we just ignore
FSpDelete(theFile);

// create the local file with the desired type and creator
myErr = FSpCreate(theFile, kTransFileCreator,
                  kTransFileType, smSystemScript);
if (myErr != noErr)
    goto bail;

// create data references for the remote file and the local file
myReaderRef = QTDR_MakeURLDataRef(theURL);
if (myReaderRef == NULL)
    goto bail;

myWriterRef = QTDR_MakeFileDataRef(theFile);
if (myWriterRef == NULL)
    goto bail;

// find and open the URL and file data handlers
gDataReader = OpenComponent(GetDataHandler(myReaderRef,
                                             URLDataHandlerSubType, kDataHCanRead));
if (gDataReader == NULL)
    goto bail;

gDataWriter = OpenComponent(GetDataHandler(myWriterRef,
                                             rAliasType, kDataHCanWrite));
if (gDataWriter == NULL)
    goto bail;

// set the data reference for the URL data handler
myErr = DataHSetDataRef(gDataReader, myReaderRef);
if (myErr != noErr)
    goto bail;

// set the data reference for the file data handler
myErr = DataHSetDataRef(gDataWriter, myWriterRef);
if (myErr != noErr)
    goto bail;

// allocate a data buffer; the URL data handler copies data into this buffer,
// and the file data handler copies data out of it
gDataBuffer = NewPtrClear(kDataBufferSize);
myErr = MemError();
if (myErr != noErr)
    goto bail;

// open a read-only path to the remote data reference
myErr = DataHOpenForRead(gDataReader);
if (myErr != noErr)
    goto bail;

// get the size of the remote file
myErr = DataHGetFileSize(gDataReader, &gBytesToTransfer);
if (myErr != noErr)
    goto bail;

// open a write-only path to the local data reference
myErr = DataHOpenForWrite(gDataWriter);
if (myErr != noErr)
    goto bail;

// start reading and writing data
gDoneTransferring = false;
gBytesTransferred = 0L;

gReadDataHCompletionUPP =
    NewDataHCompletionUPP(QTDR_ReadDataCompletionProc);
gWriteDataHCompletionUPP =
    NewDataHCompletionUPP(QTDR_WriteDataCompletionProc);

// start retrieving the data; we do this by calling our own write completion routine,
// pretending that we've just successfully finished writing 0 bytes of data
QTDR_WriteDataCompletionProc(gDataBuffer, 0L, noErr);
```



```

bail:
    // if we encountered any error, close the data handler components
    if (myErr != noErr)
        QTDR_CloseDownHandlers();

    return((OSErr)myErr);
}

```

So we've managed to use QuickTime's data handlers to provide a general-purpose network file-transfer capability that operates asynchronously, allowing us to play movies or perform other operations while the transfer is underway. Of course, there are still some refinements we might add, such as alerting the user if he or she decides to quit the application while a file transfer is in progress. We'll leave these as exercises for the interested reader.

Notice that we call the function `DataHGetFileSize` to determine the size of the remote file (which is of course the number of bytes we need to transfer). `DataHGetFileSize` may need to read through the entire remote file to determine its size, which can sometimes slow things down (since we're calling it synchronously). Some data handlers (but not all) support the `DataHGetFileSizeAsync` function, which allows us to get this information asynchronously. You might try experimenting with `DataHGetFileSizeAsync` to see if it improves performance in your particular situation.

In that vein, you might be wondering: "sure, we can use QuickTime to transfer data across the net, but is it any good? What's the performance like?" My preliminary (and admittedly unscientific) tests show that our code is in fact very good. In a few sample FTP transfers, `QTDataRef` consistently transferred files at least as fast as the latest version of *Anarchie*, a popular shareware Internet file transfer application for the Mac. Moreover, with a movie playing continuously in the foreground, `QTDataRef` took only about 10% longer to transfer the file. (And don't forget that our code works on MacOS 8 and 9, Windows, and Mac OS X!)

DATA REFERENCE EXTENSIONS

Consider now this question: if we pass a handle data reference to the function `GetGraphicsImporterForDataRef`, how does it figure out which graphics importer to open and return to us? Recall (from "Quick on the Draw" in *MacTech*, April 2000) that when we pass a file specification record to `GetGraphicsImporterForFile`, it first inspects the Macintosh file type (on Mac OS) and then the filename extension of the specified file. If neither of these inspections reveals the type of image data in the file, `GetGraphicsImporterForFile` must then validate the file data (that is, look through the file data for clues to the image type). With a handle data reference, where there is no file type or filename extension, only the validation step is possible. Unfortunately, validation is time-consuming and, alas, not guaranteed to produce correct results.

QuickTime 3.0 provided a preliminary solution to this problem by allowing us to attach a filename to the referring data of a handle data reference. (Let's call this a *filenaming extension*.) That is to say, a handle data reference is a handle to a 4-byte handle that is optionally followed by a Pascal string containing a filename. Listing 15 defines the function `QTDR_AddFilenamingExtension` that attaches a filename to the

referring data of a handle data reference.

Listing 15: Appending a filename to some referring data

```

OSErr QTDR_AddFilenamingExtension (Handle theDataRef,
                                   StringPtr theFileName)
{
    unsigned char    myChar = 0;
    OSErr            myErr = noErr;

    if (theFileName == NULL)
        myErr = PtrAndHand(&myChar, theDataRef, sizeof(myChar));
    else
        myErr = PtrAndHand(theFileName, theDataRef,
                           theFileName[0] + 1);

    return(myErr);
}

```

The filename can contain an extension that provides an indication of the kind of data in the data reference target. For instance, a filename of the form "myImage.bmp" indicates that the data consists of Windows bitmap data. For reasons that will become clear in a few moments, `QTDR_AddFilenamingExtension` looks to see whether the `theFileName` parameter is `NULL`; if it is, `QTDR_AddFilenamingExtension` appends a single byte whose value is 0.

QuickTime 4.0 provides a more complete solution to this problem by allowing us to create data reference extensions for handle data references. A data reference extension is a block of data that is appended to the referring data, in pretty much the same way that we just appended a filename to that data. The

<http://www.scientific.com>

Professional Software Developers

Looking for career opportunities?

Check out our website!

Nationwide Service

Employment Assistance

Resume Help

Marketability Assessment

Never a fee

Scientific Placement, Inc.

800-231-5920 800-757-9003 (Fax)

das@scientific.com

main difference is that, unlike the filenaming extension, a data reference extension is packaged as an atom, with an explicit type. QuickTime currently supports four kinds of data reference extensions, defined by these constants:

```
enum {
    kDataRefExtensionChokeSpeed      = FOUR_CHAR_CODE('chok'),
    kDataRefExtensionMIMEType        = FOUR_CHAR_CODE('mime'),
    kDataRefExtensionMacOSFileType    = FOUR_CHAR_CODE('ftyp'),
    kDataRefExtensionInitializationData = FOUR_CHAR_CODE('data')
};
```

A data reference extension of type `kDataRefExtensionChokeSpeed` can be added to a URL data reference to specify a choke speed (which limits the data rate of a file streamed using HTTP streaming). The other three types can be added to a handle data reference to help identify the kind of data in the target of the data reference or to supply some initialization data to the data handler. If a data reference extension is present, then the filenaming extension must also be present. The filename can be 0-length, however, in which case the filenaming extension consists only of a single byte whose value is 0.

Listing 16 shows how to add a Macintosh file type as a data reference extension.

Listing 16: Appending a file type data reference extension

```
OSErr QTDR_AddMacOSFileTypeDataRefExtension
    (Handle theDataRef, OSType theType)
{
    unsigned long  myAtomHeader[2];
    OSType         myType;
    OSErr          myErr = noErr;

    myAtomHeader[0] = EndianU32_NtoB(
        sizeof(myAtomHeader) + sizeof(theType));
    myAtomHeader[1] = EndianU32_NtoB(
        kDataRefExtensionMacOSFileType);

    myType = EndianU32_NtoB(theType);

    myErr = PtrAndHand(myAtomHeader, theDataRef,
        sizeof(myAtomHeader));
    if (myErr == noErr)
        myErr = PtrAndHand(&myType, theDataRef,
            sizeof(myType));
    return(myErr);
}
```

This code simply calls `PtrAndHand` to append an atom header to the referring data and then calls `PtrAndHand` again to append the file type (suitably converted to big-endian format).

Another way to flag the type of data in a handle is by specifying a MIME type. MIME (for Multipurpose Internet Mail Extension) is a standard protocol for transmitting binary data across the Internet. A MIME type is a text string used in MIME transmissions to indicate the type of the data being transmitted. (For instance, the string "video/quicktime" is the MIME type of QuickTime movie files.) MIME types can also be used locally to indicate the type of a file or other collection of data. Movie importers and graphics importers will look for MIME type data reference extensions to help identify the type of data specified by a handle data reference. If you are building some movie or image data in memory, you can use the

`QTDR_AddMIMETypeDataRefExtension` function, defined in Listing 17, to add a MIME type as a data reference extension. (Be sure to add a filenaming extension before adding a MIME type data reference extension.)

Listing 17: Appending a MIME type data reference extension

```
OSErr QTDR_AddMIMETypeDataRefExtension
    (Handle theDataRef, StringPtr theMIMEType)
{
    unsigned long  myAtomHeader[2];
    OSErr          myErr = noErr;

    if (theMIMEType == NULL)
        return(paramErr);

    myAtomHeader[0] = EndianU32_NtoB(sizeof(myAtomHeader) +
        theMIMEType[0] + 1);
    myAtomHeader[1] = EndianU32_NtoB(
        kDataRefExtensionMIMEType);

    myErr = PtrAndHand(myAtomHeader, theDataRef,
        sizeof(myAtomHeader));
    if (myErr == noErr)
        myErr = PtrAndHand(theMIMEType, theDataRef,
            theMIMEType[0] + 1);
    return(myErr);
}
```

CONCLUSION

As we've seen, QuickTime uses data references to find the data that it's supposed to handle. Data references can be embedded in movie files or passed to Movie Toolbox and ICM functions. So whether we're building movies or operating on them, understanding data references is crucial to doing any real work with QuickTime. Here we've learned how to work with data references to create reference movie files, play movies from RAM, and open movies located remotely on the Internet. In the previous article, we also saw how to use data references to create shortcut movie files and embedded movies. In future articles, we'll work with data references as a normal part of our QuickTime programming. So it's good that we've taken time to learn how to create and work with them.

On the other hand, data handlers are normally transparent to applications. We can use them directly for certain special purposes, such as transferring remote files to the local machine. But normally the QuickTime APIs insulate us from having to work with them at all. Typically, we can accomplish what we need by handing the Movie Toolbox or ICM a data reference and letting it communicate with the appropriate data handler.

CREDITS AND REFERENCES

Thanks are due to Chris Flick and Scott Kuechle for reviewing this article and offering helpful comments, and to Peter Hoddie for providing some historical perspective.

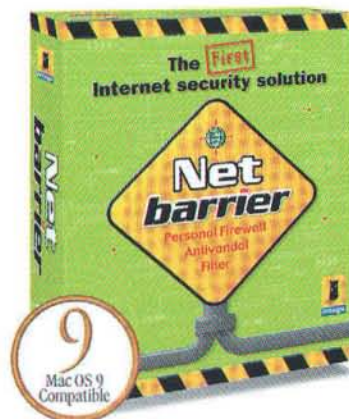
The official Apple documentation on data handlers can be found at <http://developer.apple.com/techpubs/quicktime/qtdevdocs/RM/rmDataHandlerComp.htm>. See also <http://developer.apple.com/technotes/tn/tn1195.html> for another discussion of reference data extensions.



You think the Internet is safe. Think again...



NetBarrier. The first Internet security solution for Macintosh.



All Macs connected to the Internet (dialup, DSL, cable-modem) are exposed to hackers. Whether you are a home user or a professional user, your data interests them. That's why you need a security solution that only NetBarrier can provide.

Personal Firewall

NetBarrier protects and monitors all incoming and outgoing data. A customized mode allows you to create your own defense rules, thereby offering the most secure level of protection.

Antivandal

NetBarrier blocks all attempts to break into your Mac, detects wrong passwords and logs vandal attacks for complete protection. Moreover, it has an alarm to inform you of every intrusion attempt.

Internet filter

NetBarrier analyzes data as it leaves your computer and prevents unauthorized exporting of private information such as credit card numbers, passwords, sensitive data and more...

Available @ **Developer DEPOT**

<<http://www.devdepot.com>>
Toll Free: 877-DEPOT-NOW
Outside U.S./Canada: 805/494-9797

www.intego.com



By Patrick Taylor and Sam Krishna

How to build an EOModel (or look just like one)

Back in 1996, there wasn't much competition for WebObjects. Nowadays however there are dozens of web application servers on the market, each promising that their package will make it oh-so-easy to publish data from your relational database to the World Wide Web.

If you picked some other web application server you generally had a choice of two solutions: tying yourself to a single vendor's database package (the "simple" solution) or embedding SQL into your page template (the not-so "simple" solution). If writing SQL is your idea of a fun Friday night or if you don't believe that object-oriented classes are a spectacular improvement over SQL, then you probably won't fully appreciate EOF's special charms. But even if you aren't 100% sold on the Enterprise Objects Framework, the sexy EOModeler application might still turn your head.

EOModeler, one of Apple's WebObjects development tools, provides you with a simple way to manage a great deal of the complexity associated with databases. EOModeler lets you graphically design your object/data model.

In EOModel, you replace some of your relational database terminology with object-oriented ones. However, the concepts map quite neatly.

EOENTITY

An EOEntity represents an actual class in an object model, one which generally stands for a table in a database. Fundamentally, though, an EOEntity represents some piece of information or data that must be captured and represented in your application.

EOATTRIBUTE

An EOAttribute represents a column of data in a database table. EOEntities are composed of EOAttributes. An EOAttribute has two intrinsic data types: a class type and an external type. The class type represents the object type (like a java.lang.String or an NSString) that the attribute will be as an enterprise object's (EO) instance variable. The external type represents the kind of database type (like a VARCHAR) that the attribute will be stored as in the database.

EORELATIONSHIP

An EORelationship represents a relationship between two EOEntities. There are two fundamental types of relationships: to-one and to-many. What this means is that an EO can have either a to-one relationship to a single EO of a particular type or a to-many relationship to potentially various EOs of a particular type. A few variations on both of these fundamental types exists: many-to-many (where an EO can have a relationship to many EOs of a different type, and EOs of the different type can have multiple relationships back to the original type of EOs), reflexive to-one (where an EO has a relationship back to another EO of the same type) and reflexive many-to-many (where an EO can have to-many relationships back to other EOs of the same type). We will only be covering regular to-one, to-many, and, of course, many-to-many relationships in this article.

EOEntities are Composed of EOAttributes and Connected to Other EOEntities Through EORelationships

EOModeler provides a Wizard that can generate an EOModel from a pre-existing database saving you from much of the effort of recreating an object model from scratch. How complete the object model representation of the underlying database is depends on the adaptor and the database; some adaptors like ODBC provide only basic information like entities and attributes, but not relationships or other database-specific features. You will need to manually modify the EOModel to provide the necessary missing features.

If you're familiar with Microsoft Access' graphical modeling tools, you will rather quickly pick up on the graphical modeling mode in EOModel. You aren't limited however to viewing data flows graphically, EOModeler provides developers with two other views: an outline view and a class browser view. You can view additional information about an EOEntity, EOAttribute or EORelationship with an Inspector.

One interesting thing about EOModeler is that it isn't a one-way only tool, it can generate SQL to create a database from an EOModel. For instance you could generate an EOModel from a MSAccess database and then generate the SQL necessary to recreate the database structure in Oracle 8i. Or you could design your object model from scratch in EOModeler and then using the SQL generator build a database structure. Through its adaptors, WebObjects/EOF provides developers with a remarkable degree of database independence.

AN EXAMPLE APPLICATION: PROJECTMANAGER

To truly understand EOModeler it helps to see it work in an application. ProjectManager, our example application, is intended to assist a Project Manager in allocating resources for the various projects that he or she manages in a company. One way of describing the entities would be:

Employees
Projects
Departments
Managers

Real-life Project Managers reading this article will see that we've simplified the number of entities that normally participate in a real project model. However, we can simplify this model even further when we realize that Manager is just a different type of employee and does not need to be its own entity. Further simplified the current set of entities are:

Employees
Projects
Departments

ACCESS, SIGH ...

There are several very good reasons for using MS Access in your WebObjects application. It is omnipresent, practically every business has an Access database. It is cheap especially if you already own MS Office Professional for Windows. It is easier to create a database with it than with many other RDBMS.

However, you are likely to run into some problems when using Access with WebObjects. You need to ensure that you're using a very recent version of the ODBC drivers. In earlier versions of Access, we had problems with Autonumber in applications which weren't read-only. Unlike other RDBMS, Access only passes a bare minimum of information to the EOModeler Wizard and only accepts a bare minimum from EOModeler's SQL generating feature.

This isn't to say that you shouldn't use MS Access (well ...) but it probably won't be simpler than Frontbase or Openbase because you won't have full access to all the features and automation available through EOModeler.



Having identified the entities in this application, we need to make them usable within the ProjectManager object model. Entities are named using the singular form:

Employee
Project
Department

Singular form is used in order to simplify and rationalize the naming of relationships. You will later need to use the plural form, so it's easier to name entities in singular form rather than try to figure out: "What is the plural form of 'Employees'?"

The next thing we do is create the entities in the EOModel. Generally, we have to define three things in order to create an EOEntity: the entity name, the table name, and the class name.

EOModeler's default behaviour is to assign any EOEntity's class name to EOGenericRecord, which is simply a data object that can be used to manipulate an EO's data without going to the trouble of defining a class or class behavior. In this case we'll give the Employee entity the class name Employee.

Entity Name	Table Name	Class Name
Employee	EMPLOYEE	Employee

Common practice in EOModeling is to borrow the database convention of naming tables and columns with all upper case characters. While it isn't necessary to maintain the same names, the entity, table and class names are the same for this example.

Once we've finished filling out the required information for the EOEntity, we can start defining the EOAttributes we want to track in each EOEntity. In the Employee entity, we want to track certain things:

First Name
Last Name
Employee Number
Address
Salary
Department
Manager
Start Date
Title

Valentina

object-relational database engine

The fastest database engine for the Mac OS

It operates 100's and sometimes a 1000
times faster than other systems

Valentina - scriptable DBMS..... \$49

- Includes special features for Web Developers.
- Glues for Frontier and MacPerl.

Valentina C++ SDK \$499/\$699

- Cross-platform (Mac OS/WIN32);
- bool, byte, short, long, float, double, date, time string, BLOB, TEXT, Picture.
- RegEx search, full text indexing, support international languages.
- SQL, random-access cursors.
- Multi-threading capable.
- Record locking.
- No runtime fees.

Valentina for REALbasic plugin \$199

Valentina for Macromedia Director Xtra \$199/299

Valentina for WebSiphon \$299

Valentina XCMD \$199

Make your application's database operations blazingly fast!

Order Directly **www.paradigmasoft.com**
from Our Web Site Hosted by MacServe.net

Download full featured evaluation version

Here's how we model that information:

Employee Entity

firstName
lastName
employeeNumber
employeeID
streetAddressOne
streetAddressTwo
city
state
zipCode
salary
departmentID
startDate
title

You'll notice some things right off. First, the Attributes

NAMING

Naming in an Object Model is probably the most important thing a WebObjects developer can do. If entities, attributes, and relationships are named well, a developer can develop and debug so much faster than if elements are improperly named.

Some common problems that occur in bad object modeling:

- 1) Poorly named or misnamed entities
- 2) Abbreviated names of entities
- 3) Abbreviated names of attributes
- 4) Misnamed attributes
- 5) Misnamed relationships

If you can at all avoid them, abbreviations that aren't part of the common vocabulary (like ID) shouldn't appear in your EOModel. Well-named entities, attributes, and relationships make life easier for the developer—so much so that applications that take months to develop could shave several weeks off with strong naming. Good naming is particularly important if you share responsibility for an application with other developers or if someone else may someday inherit your code. **MT**

start off lower case but are intercaptioned since spaces aren't used between words. The names are descriptive to assist in recognition. You can name entities with acronyms or nonsensical names, but you're only punishing yourself. You'll also notice that there isn't always a one-to-one mapping between the identified items and the actual entity.

For our application, Address is more complex than what can be represented usefully in a single attribute — because of certain postal conventions, modeling it usefully requires that we use several smaller attributes, like streetAddressOne, streetAddressTwo, etc. (these address rules apply in the

United States but may differ in other countries). You will also notice that the employee's department only appears as a departmentID. We have a departmentID instead of actual department attributes because in our application the employee's department information is actually a relationship that an Employee has with a Department. There are two reasons for choosing this: it conserves resources by storing an ID (32 or 64 bits long) rather than multiple potentially redundant entries (16 bits per Unicode character).

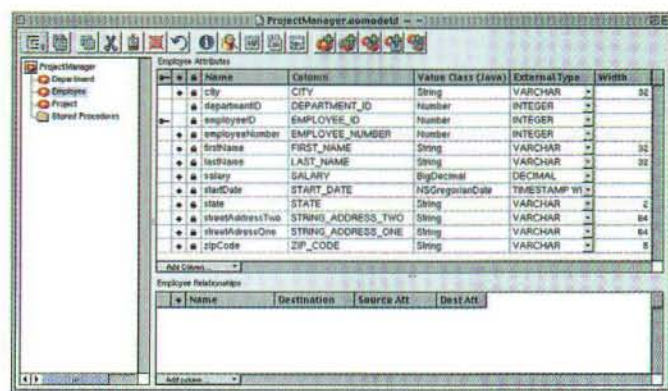
When you model a particular attribute, you need to define:

attribute name
attribute class value
attribute external type
attribute column name
attribute width (if defining an attribute that tracks a string object)

We also define whether or not an attribute is a class property. Class properties are identified with a small diamond to the left hand side of the attribute name. When attributes are identified as class properties they can be manipulated in code. Attributes that are not marked as class properties are handled automatically by the EOF subsystem.

A word on class properties: any primary or foreign keys should not be marked as class properties (ie missing the small diamond). The EOF subsystem should manage any primary or foreign keys automatically. This way a lot of the magic of EOF is used to deal with key generation with the database.

After all the attributes are defined the EOEntity should look like this:



Name	Column	Value Class (Java)	External Type	Width
City	CITY	String	VARCHAR	32
DepartmentID	DEPARTMENT_ID	Number	INTEGER	
EmployeeID	EMPLOYEE_ID	Number	INTEGER	
EmployeeNumber	EMPLOYEE_NUMBER	Number	INTEGER	
FirstName	FIRST_NAME	String	VARCHAR	32
LastName	LAST_NAME	String	VARCHAR	32
Salary	SALARY	BigDecimal	DECIMAL	
StartDate	START_DATE	NSDate	TIMESTAMP	
State	STATE	String	VARCHAR	2
StreetAddressTwo	STREET_ADDRESS_TWO	String	VARCHAR	64
StreetAddressOne	STREET_ADDRESS_ONE	String	VARCHAR	64
ZipCode	ZIP_CODE	String	VARCHAR	8

EORELATIONSHIPS

EORelationships represent relationships from one EO to another EO (or groups of EOs). There are two fundamental types of relationships: to-one and to-many. Since our mythical company only assigns employees to a single department, Employee should have a to-one relationship to Department.

REALITY-BASED OBJECT MODELING

A well-named object model will do neither a developer nor a client any good if the object model does not represent the reality of a business domain as accurately as possible.

Unfortunately there aren't a lot of positive indicators for how reality-based an object model is. However, there is a set of negatives you can watch out for:

- 1) Primary or foreign keys are class properties

There should never be a reason for this. This is probably the number one reason why object models in WebObjects get excessively complex and become difficult to maintain.

- 2) Excessively deep relationship paths

An example of this would be traversing this kind of relationship chain:

Husband->Wife->Sister->Mother->Brother when all you're looking for is:

Husband->Wife->Uncle

Poor models will have an excessively deep relationship graph for a simple piece of information.

- 3) Entire to-one relationships are flattened into a table

For example, if a Project was limited to only two employees per project, you might see a Project poorly modeled like this:

```
Project
name
clientName
employeeOneFirstName
employeeOneLastName
employeeTwoFirstName
employeeTwoLastName
```

When you start seeing a lot of duplication of data between tables, it may be time to create a relationship.

- 4) Excessively difficult-to-explain-or-name relationships and entities

This is a little harder to describe, but you'll probably know it when you see it. For example, you may see or create an EOEntity that you cannot name well after thinking about it for 20-30 minutes. Or you may create a Join Entity that has a to-many relationship to a third entity that neither of the entities on either side of the many-to-many relationship needs.

This list is by no means complete but keeping these factors in mind should help you in the always difficult task of modeling. Keep practicing at it: a well-structured and accurate Object Model with a strong naming convention is worth its weight in gold. Good luck.



So Many Choices!

AnthroCarts!

Modular, Mobile and Strong.



You tailor your own AnthroCart. With so many models, and lots of accessories, you can make it fit anyone and any space. And change it when your needs change. Plus you can't beat its Lifetime Warranty. Suit yourself! Come visit our web site or call us!

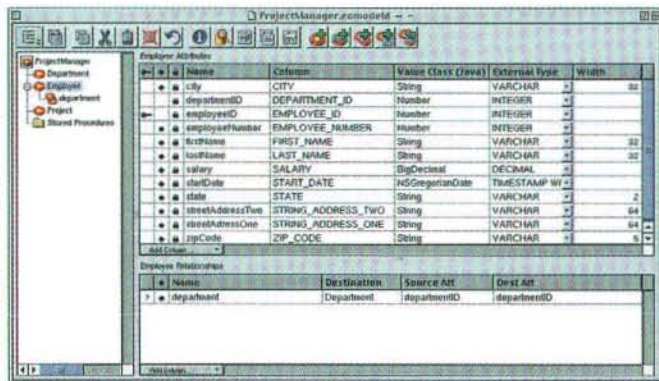


Call us for a free catalog at
800-325-3841

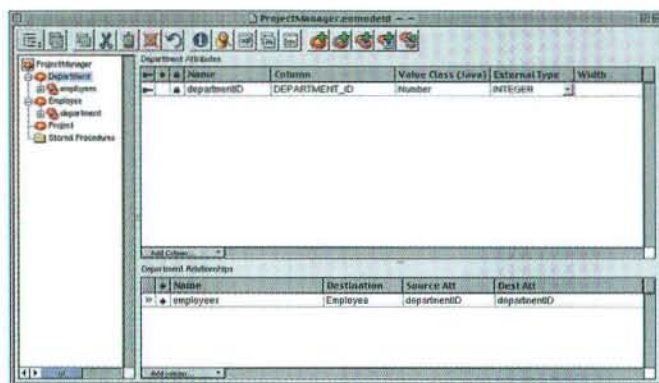
6:00 AM to 6:00 PM PST M-F

www.anthro.com

Anthro Corporation® Technology Furniture® Since 1984.



Our mythical company has multiple employees in each department so Department will have an inverse to-many relationship back to Employee.



Department and Project entities must track the following types of information:

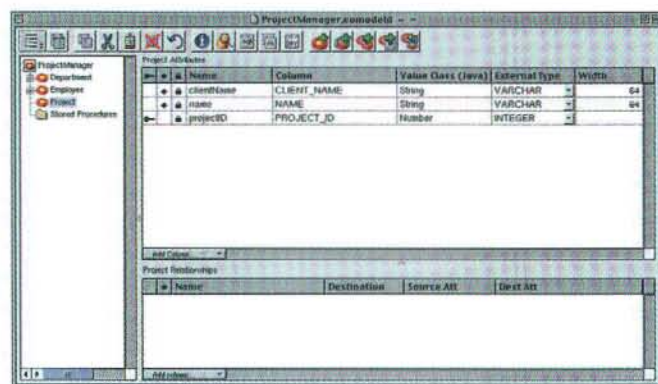
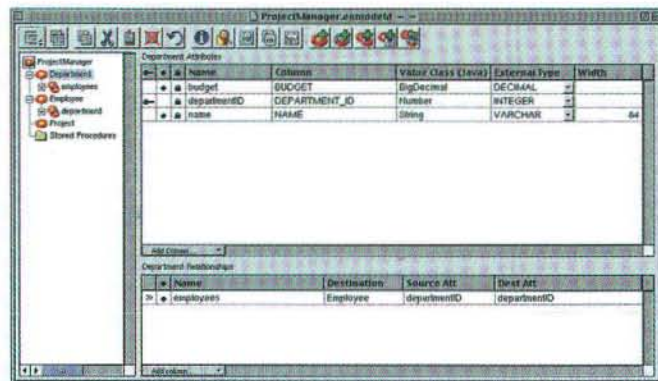
Department

name
budget

Project

name
clientName

Here are pictures of the fully modeled Department and Project entities without relationships:

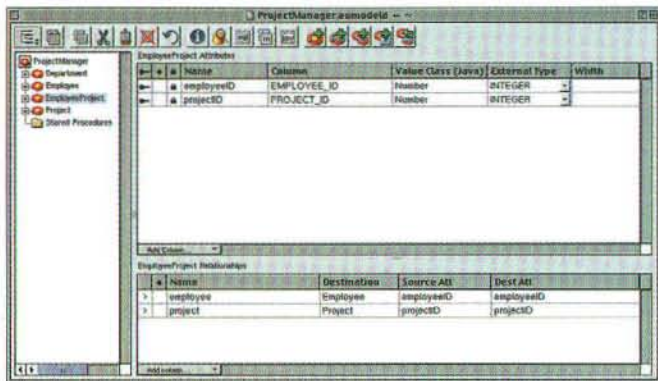


Employees aren't limited to working on a single project and each project will tend to have several employees assigned to it. Because of this we need to create a more complex type of relationship between Project and Employee than those we've discussed. The type of relationship that represents these complex systems is called a "many-to-many" relationship. Unlike to-One and to-Many relationships, Many-to-Many relationships can't be represented just by an EORelationship.

What is required to create a Many-to-Many relationships is a 'Join Entity'. A Join Entity is a special type of EOEntity that links two entities to each other in a many-to-many relationship. The Join Entity stores the Primary Keys for each of the two joined entities. Following common practice, we'll name the join entity EmployeeProject to show its place in the relationship between Employee and Project.

EmployeeProject will only have two attributes and two relationships. The two attributes will be employeeID and projectID, respectively. The two relationships will be named employee and project, respectively. Here is what the fully modeled EmployeeProject join entity looks like:

Want to write for MacTech?
Download a writer's
kit from the web at
www.mactech.com

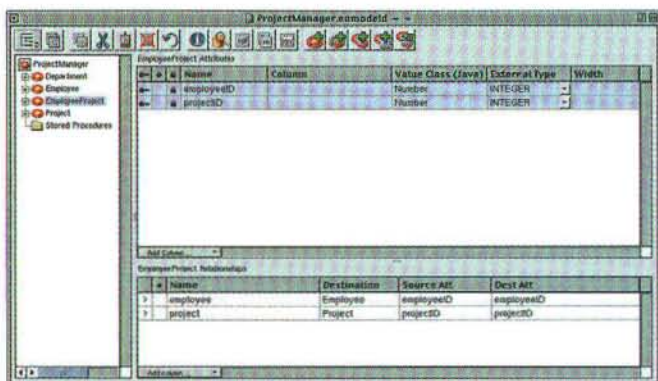
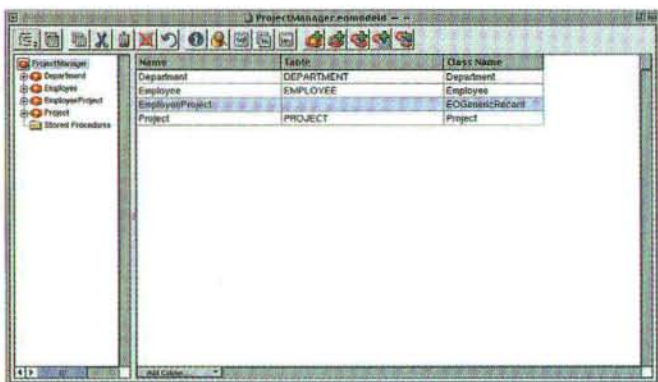


SHORTCUT!

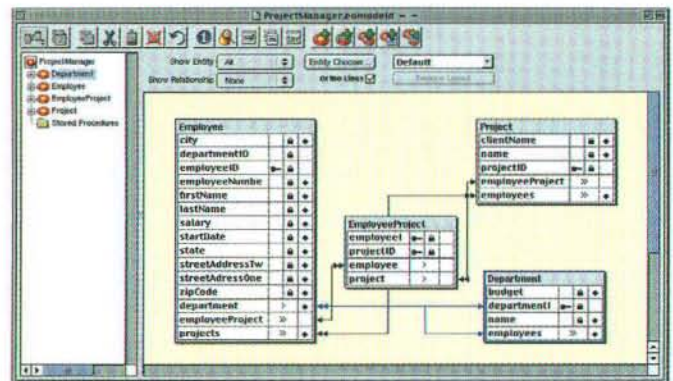
There is a shortcut in EOModeler which simplifies the creation of many-to-many relationships. Here are the steps to the shortcut:

- 1) Select the two entities you wish to join
- 2) Select from the Properties menu, 'Join in Many-to-Many relationship' or select the Many-to-Many icon on the toolbar.

After the Join Entity has been created, you will need to fill in the table name and the attributes' column names. It will look something like this:



Once modeling is complete, here is a diagram of what your model should look like with entities, attributes, and relationships:



You have now built an object-relational model without writing a single line of code. No SQL was needed and, for the most part, you could use any database you wanted. The database portion is often the most difficult and painful part of the development process.

MT

BMS

THE LAW OFFICE OF BRADLEY M. SNIDERMAN

23679 Calabasas Road #558 • Calabasas, CA 91302
Tel: 818/222-0365 • Fax: 818/591-1038

Got Software?

Need help safeguarding your software? If you're developing software, you need your valuable work protected with copyright and trademark registration. Then, when you are ready to sell it, you can protect yourself further with a licensing agreement.

I am a California Lawyer focusing on Intellectual Property, Corporate, Commercial, and Contract Law, as well as Wills & Trusts.

Please give me a call or an e-mail. Reasonable fees.

The Law Office of Bradley M. Sniderman



Visa
Master Card

Discover
American Express

E-mail: brad@sniderman.com • Internet: www.sniderman.com

By Kas Thomas

How to Harness the Power of Associative Arrays

A widely underutilized technique can go a long way toward making your code smaller, more reliable, and easier to maintain, whether you program in C, Java, or JavaScript

Fostering reliability, maintainability, compactness, and good performance in code has been a constant quest for programmers and language designers over the years. It's rare that one technique can give you all of the above benefits, concurrently. But intelligent use of associative arrays can do that. If you haven't yet tapped into the power of associative arrays, you might want to give the issues involved some thought. The issues are widely applicable to a variety of programming tasks, cutting across all major languages.

One thing's for sure. If you use a lot of switch statements in your code (to "case out" user-selected actions, for example), you're sitting on a great opportunity to improve the reliability and maintainability of your code; just follow along and see.

WHAT IS AN ASSOCIATIVE ARRAY?

An associative array is a collection of data values in which access to any particular value is obtained via a (single) key. Generally speaking, it's a one-to-one

mapping of data entities ("values") to keys, such that no two values map to the same key (although two different keys could, coincidentally, contain the same value). For example, in an array of TV listings, you might very well find that two different channels are showing the same episode of Gilligan's Island at 4:00 a.m. Tuesday, but you would never find Channel 2 listed twice, showing two different shows in the same time slot.

In a larger sense, associative-array mapping is an application of set theory, and languages that have a robust collections package will invariably implement some form of associative-array mapping.

For a quick example of an associative array, consider the hypothetical entity in **Figure 1**.

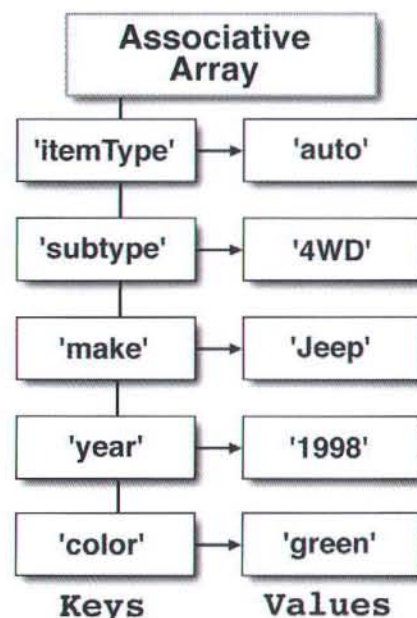


Figure 1. A hypothetical associative array.

In this example, we have an anonymous (nameless) array of tuples shown as keys mapped to values. The itemType key, for

Kas Thomas has been programming in C and assembly on the Mac platform since before Windows existed. By day, he is a Senior Technical Writer for Silverstream Software (a maker of app-server software). You can reach him in care of editorial@mactech.com.

example, maps to the value 'auto'. If this were a conventional (ordered) array, we would say that the zeroth item in the array is the (string) value 'auto', the first item is '4WD', etc.

For the array shown in Figure 1 to be useful, we need to be able to hold a reference to the array in a variable, and we need to have a syntax for obtaining the value of each member, given the value of the corresponding key. It would also be nice if we could find out how many items are in the array (since it might be arbitrarily large) and to know how to iterate through the members. With an ordered array, you can step through the members in sequence. But in an array like this one, where the keys aren't integers, how do you enumerate the array's contents?

ASSOCIATIVE ARRAYS IN JAVASCRIPT

Core JavaScript (or ECMAScript) has a convenient syntax for dealing with constructions such as this. (Even if you're not a JavaScript programmer, bear with me here, because I'll be discussing other languages in a moment.) In the world of object-oriented programming, an arbitrary collection of data entities is often called an object. In JavaScript, a generic object is, in fact, an associative array (whose members can be references to functions, in which case those particular members are called methods). In the OOP world, dot notation is commonly used to "index into" an object. For example:

```
var myObject = new Object();
myObject.itemType = 'auto';
myObject.subtype = '4WD';
myObject.make = 'Jeep';
myObject.year = '1998';
myObject.color = 'green';
```

In JavaScript, this is an entirely acceptable way to implement the object suggested by **Figure 1**. An equivalent syntax that uses literal notation is:

```
var myObject = { 'itemType' : 'auto',
                 'subtype'  : '4WD',
                 'make'     : 'Jeep',
                 'year'     : '1998',
                 'color'    : 'green' };
```

But are we justified in calling this an associative array? Yes, we are, because it turns out we can also use array notation (square brackets) in JavaScript to refer to object properties:

```
var myObject = new Object();
myObject['itemType'] = 'auto';
myObject['subtype'] = '4WD';
myObject['make'] = 'Jeep';
myObject['year'] = '1998';
myObject['color'] = 'green';

if (myObject['itemType'] == 'auto'
    && myObject['make'] == 'Jeep') // true
    myObject['subtype'] = '4WD'; // assign '4WD' to myObject.subtype
```

In JavaScript, square brackets can substitute for dot notation when referring to object properties. The only proviso is, the enclosed property name must be supplied as a string literal (as shown above). This turns out to be an extremely handy syntactical construct, because it means we are not limited to one-word property names. We can now consider constructions like:

```
myObject['extended warranty'] = true; // legal
myObject['original delivery date'] = '1/1/98'; // legal
myObject.'original delivery date' = '1/1/98'; // ERROR!
```

FUNNELWEB
VERSION 4



Better, Stronger, Faster

Intelligent Web site Monitoring and Analysis Software

Speed, intuitive user interface, accuracy and in-depth analysis have always made Funnel Web the intelligent choice for Web site analysis.

Now includes pdf output, incremental analysis, cluster analysis and streaming media reports.

Better, Stronger, Faster

Iterating through the contents of an associative array (or the properties of an object) would seem to present a problem, since the contents aren't necessarily ordered and can't be "indexed into" numerically. In JavaScript, however, there is a very handy loop syntax for doing this:

```
var bigstring = new String();
for (var i in myObject)
    bigstring += i + "\n";
```

The above code builds a list of all the property names in myObject (separated by newlines), concatenating them into a single big string.

PERL

Perl has a built-in associative-array construct called the hash. The syntax for creating a hash looks like:

```
%myObject = ('itemType' , 'auto',
             'subtype' , '4WD',
             'make' , 'Jeep',
             'year' , '1998',
             'color' , 'green');
```

Notice that commas are used throughout the pairs list, rather than the more readable JavaScript technique of putting colons between keys and values (using commas to separate complete tuples).

To access a hash value in Perl, you use a notation that looks like:

```
$myObject{ 'color' } = 'red';
```

Notice the seemingly inconsistent use of the dollar sign (instead of the percent sign) and curly braces (instead of parentheses) for accessing individual values of a hash. This is a standard Perl idiom. The percent sign refers only to complete hashes.

JAVA

With the advent of the Java 2 platform, the java.util package now has a powerful Collections Framework (not to be confused with JGL, the Generic Collections Library for Java by ObjectSpace, which was first-to-market and was widely distributed in IDEs before Sun came out with the J2SE Collections Framework), encompassing sorted and unsorted sets, maps, and lists, plus iterators for same, in thread-safe and non-thread-safe versions. Prior to Java 2, the language had Vector and Hashtable classes to accommodate this need, but those classes have been relegated to legacy status. Today, if you need an associative array, you call on the HashMap class, which implements the Map interface:

```
Map map = new HashMap();
map.put("itemType", "auto");
map.put("subtype", "4WD");
map.put("make", "Jeep");
map.put("year", "1998");
map.put("color", "green");
```

For sorted maps, you can create a TreeMap (using a Map as input, if necessary), which stores its elements in an always-balanced tree.

C AND C++

In C++, the Standard Template Library provides rich support for collections idioms of all kinds. The map is an associative container that accomplishes the kind of functionality we've been talking about:

```
map<string,string> myMap;
myMap["itemType"] = "auto";
months["subtype"] = "4WD";
// etc. etc.
```

To use it, be sure to #include <map>. See any good reference on STL for further details.

ANSI C, on the other hand, has no built-in support for associative arrays. You can craft your own support, of course, with a little effort. But how? As you might guess from the repeated use of the strange term "hash" in this context, implementing an associative array from scratch requires the use of hash techniques.

What, exactly, is a hash? A hash is basically a mathematical function in the canonical sense: a "black box" that maps domain values to range values, with the requirement that any one domain value can map to no more than one range value. This is exactly the mapping behavior we're looking for, of course.

But what does a hash function really look like? The answer is, it can look like whatever you want it to look like, or (more commonly) whatever you need it to look like. If that sounds a bit vague, that's because hash techniques fall into a curious category of computational heuristics with a long background in computer-science lore. In the old days of limited machine resources, you'd use hash techniques to maximize performance while minimizing storage requirements. Good hash functions weren't often published, and if they were, their workings were rarely self-evident.

A HASH-CODE EXAMPLE

A quick example may help. Suppose you are implementing an exception dictionary for a spellchecker: i.e., a special dictionary to store user-entered words. What you want is that when a user types a word that is not in the regular dictionary, the exception dictionary can be consulted; and at that point, you simply need to know whether the word exists in that dictionary. (If not, you flag the word as a possible misspelling and notify the user.) You have a maximum of 5 Kbytes of storage available for this task, because you're running on a 1970s machine with severely limited resources.

A typical answer to this challenge would be to insert exception words into a linked list, and do a lookup by traversing the list. With 5K of storage and an average word length of five characters, you could store as many as 1,000 words in the list (more than adequate for most exception dictionaries). But lookups will be time-proportional to the number of words in the list, and there will be significant overhead in terms of list management, because each time a new word is entered in the list, you have to be sure it doesn't already exist.

This approach gets a D for ingenuity.

A bright young programmer in your department comes to you at this point and says "Wait! I have a better idea." He explains that if you keep the linked list sorted alphabetically, you can speed lookups because they will occur in time-proportion to the log of the number of entries. List-maintenance overhead is reduced as well, although storage requirements haven't changed.

This represents a significant improvement. Let's give it a C+ (or maybe even a C++) for effort.

At this point, a consultant walks in the door and offers to solve the problem for you in such a way that up to 40,000 words can be stored in your exception dictionary and lookups are instantaneous, with zero table maintenance. You say to him "Thank you, but that's clearly impossible. Don't let the doorknob hit you in the butt on the way out." He goes straight to your competitor, implements the solution, and eventually the competitor puts you out of business. You end up face-down in the gutter, next to an empty bottle of Ripple.

How did the consultant do it? First of all, since you are only concerned with whether a given word has an entry (true) in the dictionary or not (false), you're really looking for a binary (Boolean) answer to a question; hence your 5K dictionary can really store 40K lookups (assuming 8-bit bytes, of course). The dictionary needs to be implemented as a bit table.

To implement a direct lookup of words, we resort to the following heuristic. To store a word in the dictionary, we submit the word string to a hash function, which in turn converts the word from a string to a number in the range of zero to 5K-minus-one. We then index into the bit table at the offset thus obtained, and store the word by flipping the bit "on." (A zero bit means the word doesn't exist in the table.) To look up a word and see if it exists in the exception dictionary, we simply hash the word and do a direct lookup. If the entry in the table at that lookup is true, the word exists. If not, it doesn't. Problem solved.

Finally, an A+ solution!

HASH FUNCTIONS

But we still haven't resolved the question of what a hash function looks like. Earlier we merely said that it could look like whatever it needs to look like. Let's take the exception-dictionary example we just discussed. How many bits' worth of information does a 5-character string really hold? If the string represents an English word, each character can have one of 26 possible values (except that the first character can be upper or lower case; hence 52 possible values). Storing 26 possibilities requires 4.7 bits; 52 possibilities requires 5.7 bits. The average word therefore requires $5.7 + 4 * 4.7 ==$ just under 25 bits of bandwidth. Every 5-character English word can therefore be converted (mapped directly and unambiguously) to a 25-bit number. Which is to say, a number between zero and 33,554,431.

Clearly, if we had 32 million bits (4 megabytes) of storage available, and words could never be longer than five characters, you could convert words directly to binary numbers and use the numbers as offsets into a bit table.

Always Thinking

Professional Macintosh & Internet Development

Always Thinking's professional developers will help you meet your Macintosh and Internet deadlines! So if you're...

- on a tight deadline and need additional talent
- losing valuable development time debugging
- having trouble finding good developers

... **Always Thinking's** team of experienced programmers will provide you with a timely and affordable solution.

We deliver more than code — a complete project. Our software engineers work with you to:

- Create clear, solid project specifications
- Design and develop your application or web site
- Tune and optimize your software's performance
- Thoroughly test your application or site
- Completely document your project
- Provide training to your team

Commercial Product Development

Do you have an exciting idea for an application? Turn to **Always Thinking** to make it a reality. We have firsthand experience developing and shipping award-winning commercial applications for our clients and our own Thinking Home, a 2000 Apple Design Award winner.

Web Site Design & Development

Get a sound e-commerce system tailored for both your immediate needs and long-term growth. Our engineers can develop the Internet applications to transform your company into an e-business.

Successful web sites are more than graphics and code. We have the Internet marketing know-how to ensure your site is an effective business tool.

Realize substantial savings by moving to online pre-sales information, ordering and support.

Tell us about your project, toll-free

(800) 252-6479

(703) 478-0181 x103



Always Thinking, Inc.
27 James Byrnes Street
Beaufort, SC 29902

www.alwaysthinking.com sales@alwaysthinking.com

Unfortunately, in the real world, words are not constrained to five characters in length and four megabytes of table storage are not always available. In fact, our example calls for no more than 5 Kbytes (40Kbits) of storage.

Why not take the bottom 15 bits of each word (discarding the rest) and use that as an index into a 40Kbit table? The answer should be obvious. Words often have the same endings. Taking the last 15 bits of 'eating' and 'drinking' would result in both words contending for the same spot in the exception dictionary. Totally unsatisfactory.

The crux of understanding how a hash solution works in this kind of situation is to think long and carefully about the problem. If you think about it, you will realize that the ASCII values in words are not at all randomly chosen. The composition of word strings in English is constrained by spelling rules, which in turn are determined, in part, by constraints on the speakability of syllables (i.e., the geometry of the human voice apparatus). The average English word may consume only 25 bits of bandwidth, but there most certainly are not 32 million different 5-letter words in the English language! The actual number of different 5-letter words you might encounter might only be in the low thousands. Certainly, in a spellchecker context, 5-letter "exception" words (words that are not commonly found in English) would likely be numbered in the low hundreds, if that many.

Careful analysis of the problem space should convince you that what we're really dealing with here is a sparse data set. We will be mapping a few hundred exceptions (or at most, a few thousand) into a 40,000-bit table. All we need is a way to convert words to numbers that map well across the available domain.

Suppose we design a hash function in C as shown in Listing 1 (where *ch* is a pointer to the word string).

Listing 1: hash()

```
hash()
unsigned int hash(ch) {
    unsigned int hashvalue = 0;
    if (!*ch) return 0; //sanity
    do {
        hashvalue += *ch++;
        hashvalue *= 13;
    } while (*ch);
    return (hashvalue % TABLESIZE);
}
```

This hash function adds the ASCII value of each character in the string, one by one, to a temporary variable. But in between each addition, the variable is multiplied by the magic number 13, which (in binary terms) is equivalent to a left-shift by $\log(13)$ bits. (Prove to yourself that this is so.) What's the significance of 13? Nothing. It simply works. Might some other number work better? Yes! That's the magic of hash functions. Often, you can "tune" them to make them work better.

What does "work better" mean? It means you have fewer situations where two different character strings hash to the same numeric value. When two data objects hash to the same value, it's called a hash collision. Collisions are usually undesirable. In our exception-dictionary example, it could mean that a nonsense word maps to the same dictionary location as a perfectly valid word!

DEALING WITH COLLISIONS

There are various ways of handling collisions. The most common way (not for bit tables, but for tables where actual string information might be stored) is to fork a linked-list node at the point of the collision. Another method is to keep parallel tables, but with different hash functions for each. An "entry" in the dictionary would actually consist of an entry in each table. A lookup would verify that Hash 1 produces a positive result from Table 1 and Hash 2 yields a positive result from Table 2. Prove to yourself that (given suitably designed hash functions) if a collision happens in the first hash table, it's extremely unlikely that a collision also occurs, simultaneously, in the other hash table; therefore, the overall collision rate is the product of two small numbers (the individual failure rates of the hashes).

In some situations, collisions can safely be ignored; the "collider" overwrites the old table entry, and that's that. This can actually be desirable in instances where data entities are expected to regularly go out of scope. (In this case, you have what is commonly known as a "weak" associative array). In the spellchecker example, it's debatable whether collisions are important. No spellchecker is ever perfect. If the odds of a misspelled word hashing to the same spot as a legitimate word are less than, say, one in five thousand, would the user really notice? Would he care?

HASH TUNING

The fascinating thing about hash functions like the one above is that they can be tuned for better collision performance (lower collision rates). What makes this fascinating is the fact that, because the problem is so poorly defined (mathematically), an analytical solution is usually out of the question. Therefore, tuning must be done empirically: try a tweak, measure the result. Try another tweak; measure the result. Repeat until happy.

You might want to write a short program that takes a text stream (a document) as input and maps the "vocabulary" to a hash table, measuring collision rates in the process. Find out how collision rate varies with load factor (table consumption) for a given hash function. Then try changing the function in some way, and see how the collision performance changes.

The example hash function given in Listing 1 is actually a surprisingly good hash function for most purposes. With a driver program (such as HashMule; see further below), you can prove to yourself, empirically, that this modest-looking function works much better (i.e., produces many fewer collisions) than the same function with the number 12

substituted for the number 13. (This is why I called 13 a magic number.) In fact, when I ran a quick test of Listing 1 using the Declaration of Independence as the source text (hashed into a table with room for 1024 words), I found that changing the magic number 13 to 12 caused collisions to more than double.

In general, even numbers produce poor results, and non-prime odd numbers (while better) are not as good as primes.

When writing a hash function, you will always need to constrain the final output to a certain range (in our example, 5K). Be aware that, for tuning purposes, it can matter a great deal what the "modulo" constraint is. Again, it turns out that a prime number here will improve performance. So make your lookup table an odd size! (But don't take my word for it. Write a test program and prove it to yourself.)

A COMMON HASH MISCONCEPTION

A common misconception is that hash functions are designed to distribute values randomly over a given range. You should convince yourself that this is not true. After all, if hashing were this easy, every hash function would simply return a random number (converted to integer form and scaled to fit the desired range). The ideal hash function is usually one that distributes keys evenly across a range, with no collisions. Random numbers tend to cause collisions, at a rate equivalent to the hashtable load factor. (That is, when the lookup table gets half full, each new entry into the table has a 50:50 chance of generating a collision.) A good hash function can do much better, because hash functions can be constructed to remap data non-randomly. And in fact, this is exactly what you are trying to achieve. You are trying to map a sparse, non-random data set to non-random locations in a table: locations that spread out evenly and don't overlap. If you know a thing or two about the data before you begin, you can make intelligent decisions about how to craft the hash function.

Consider a list of 500 arbitrarily chosen unique words selected from Webster's dictionary at random. Imagine that you are trying to map these words into a hash table that has enough space for all 500 words, but only enough space. (But you don't necessarily know in advance how many words there are.) Can it be done? Of course. You just need to develop a hash function that maps table positions to the words' alphabetical ranking. If you can analyze the word list in advance, this is a piece of cake. If you don't have advance access to the words, but your hash object is free to reorder table data on the fly, again it's not a hard problem, because you can insert words in alphabetical order and reorder as necessary. Even if you know nothing about the words (except that they are words!), you can easily come up with a hash function that does a better job of mapping them to a table (i.e., with fewer collisions) than



BLUE DOG
**A New Breed of
Application
Service
Provider**

Get

Your App



on the

WWweb



Fast



www.thinkDog.com

**The Premier WebObjects
ASP**

mere random dart-throwing.

There's no magic procedure for constructing a good hash function. Each situation is different. What might be a great function for one set of conditions could very well be terrible under another set of conditions. The trick is to understand your data set (and what's unique about it) and understand what it is you are trying to do: map members of the set onto intervals that don't overlap.

PUTTING ASSOCIATIVE ARRAYS TO WORK

With a hash function approach in C, we haven't quite achieved the `table['string']` lookup ease of other languages, but we've approached it quite closely, because we can instead do:

```
lookup = table[ hash('myItem') ];
```

But the greater question, at this point is: How can you put associative arrays to work? One example is well-known: In compilers, symbol tables are usually accessed via a hash function that computes a likely place to begin a serial search. Since symbol-table lookup is a very frequent occurrence in compilers, the performance of the hash function (both in raw execution speed and in collision rate) can have a dramatic effect on overall compiler performance.

But what if you're not a compiler writer? In that case, you might want to consider a possible application that applies broadly across a variety of programming tasks in a variety of languages. It involves 'case' statements.

THE CASE AGAINST CASE STATEMENTS

Most high-level languages support a switch construct that allows options to be associated with desired actions. In C:

```
switch( option ) {  
    case OPTION_A : do_something();  
                    break;  
    case OPTION_B :  
    case OPTION_C : do_something_else();  
                    break;  
    // ...more options processing  
    default : do_whatever();  
}
```

Usually, the cases must refer to hard-coded values. Fall-through occurs by default, which means that if you leave a break statement out (such as with `OPTION_B` above), execution continues with the next case. The original intent of the switch construct was to keep programmers from having to resort to long chains of if/else actions (while leaving code readable). But hindsight has shown switch/case syntax to be a notably poor syntax design in a number of respects. First of all, it doesn't save any typing (over if/else actions). In fact, switch blocks are usually quite long,

because in instances where there are only a couple of cases, programmers tend to collapse everything to a few if/elses. The opportunity for typos is high in long code blocks. Also, the default fall-through behavior of switches is known to be a bad design decision, because it's not the default behavior most programmers are looking for, most of the time. (Some years ago, Sun analyzed its C compiler source code to see how often the default fall-through path was actually used. It turned out the Sun ANSI C compiler's front end had 244 switch statements, averaging seven cases each. Fall-through occurred in only 3% of cases.)

There is also a performance issue, in that the compiler must (because of the possibility of fall-through) evaluate each case in turn, up to the first match. But there's an even bigger problem, architecturally: Switch statements require you to intermix code and static data. This leads to poor maintainability and a host of other ills. It would be better, obviously, to segregate code from static data so that when the data needs to be revised periodically, it can be edited without changing any code.

The answer should be obvious by now: Convert switch blocks to associative arrays. Each case can be used, after all, as a key to find the associated action. (If the associated action is a block of code, then the array need only store a pointer to the appropriate function or macro.) In this way, you can collapse the entire switch block down to one line of code, which executes instantly.

Consider the common situation, in CGI/forms programming, of retrieving a user selection from a drop-down menu and taking action on it. Usually, you have something like this:

```
var userSelection = element.options[i].value;  
  
switch(userSelection) {  
    case 'Kansas' : // JavaScript allows string cases  
                    do_this();  
                    break;  
    case 'Ohio' :  
                    do_that();  
                    break;  
    // etc.  
    default : whatever();  
                    break;  
}
```

With an associative array, you can separate data from code and reduce all actions to a single expression. In JavaScript:

```
// build a lookup table as an Object:  
var lookupTable = {  
    'Kansas' : do_this,  
    'Ohio' : do_that,  
    // etc.  
}
```



```
// execute action based on lookup:  
lookupTable[ userSelection ] ( ) ;
```

An entire series of if/else statements (or a big, long switch statement) is thus collapsed to one expression, which does one lookup, then vectors to an action procedure. Code is segregated from data, making maintenance easier and less likely to introduce bugs. Performance is improved, many lines of code are reduced to one, and readability doesn't suffer a bit. The best of all worlds. (Just don't tell your competition.)

SAMPLE APP: HASHMULE

For this article, I wrote an educational utility for creating, studying, and comparing the collision rates of hash functions. The app, called HashMule, is actually a PDF form incorporating about 150 lines of JavaScript. It is designed to run under Acrobat Reader version 4.0 or 4.05. You can download HashMule from <http://www.acroforms.com/archive/HashMule.pdf>, or by FTP from the Mactech website.

One of the things HashMule allows you to do is enter hash functions, then execute them against text in real time. A "Hash It" button on the form will hash every word of a given text block into a hash table, then report statistics such as table load factor and hash collision rate. By making small changes in the hash function and hitting the "Hash It" button, you can see how the collision rate changes as you change "magic number" values or hash-function logic. The text of the Declaration of Independence is included in the form as a sample text block.

FOR FURTHER INFORMATION

Associative arrays are such a staple of the Perl language (where they are called hashes) that you'll find them discussed in detail in every decent Perl book and online reference.

Few JavaScript references spend adequate time discussing that language's built-in associative array capabilities. David Flanagan's *JavaScript: The Definitive Guide* (O'Reilly) is one book that does.

An outstanding introduction to the Java 2 platform's Collections Framework can be found at <http://developer.java.sun.com/developer/onlineTraining/collections/Collection.html>.

There are many good books on STL (the Standard Template Library for C++, which has a rich collections library). It's difficult, however, to find good online information on STL maps. One worthwhile entry point for STL info in general, and information on <map> routines in particular, is http://www.dinkumware.com/htm_stl/index.html. Silicon Graphics, Inc. also maintains a very detailed online STL reference: see http://www.sgi.com/Technology/STL/table_of_contents.html. But bear in mind that the SGI implementation of STL contains SGI-defined extensions as well as standard C++ templates.

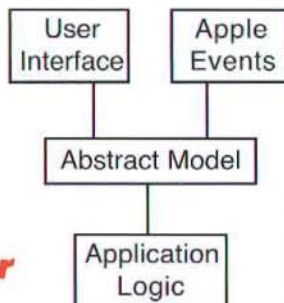
There are (IMHO) really no good books, at this point, and precious few decent online tutorials, on hash techniques. For instructive online examples, you should search the Google search engine using keywords "hash" and "cryptology." (One paper that's worth a careful look is <http://www.cs.cmu.edu/~dst/CP4break/>, "The Breaking of CyberPatrol 4," by Eddy L. O. Jansson and Matthew Skala.)

Fight Boredom

Let's face it: Much of programming is boring and repetitive. Well, that's where the right tool can save days, weeks, or even months of your valuable time.

AppMaker Your Assistant Programmer

AppMaker makes it faster and easier to make an application. It's like having your own assistant programmer. You point and click to tell AppMaker the results you want, then it generates "human, professional quality code" to implement your design.



Model-View-Controller

AppMaker's generated code uses the MVC paradigm. It separates the user interface from application logic, making code easier to write. You deal only with abstract data; AppMaker takes care of the user interface.

Scriptable Applications

AppMaker generates the 'aete' resource and generates code to access your data (Properties and Elements in the Apple Event Object Model) and to handle Events.

Just \$199 from www.devdepot.com

B•O•W•E•R•S
Development

P.O. Box 929, Grantham, NH 03753 • (603) 863-0945 • FAX 863-3857
bowersdev@aol.com • <http://members.aol.com/bowersdev>

By Daniel Jalkut

A Select Few...

Taking advantage of Apple's standardized Type Selection API

INTRODUCTION

A feature frequently overlooked by new Mac OS users is the ability to select items from a list by typing part of its name. Apple calls this functionality "type selection." Seasoned Mac users are accustomed to type selection and navigate rapidly through Finder hierarchies and StandardFile dialogs without ever letting their hands leave the keyboard. When a user discovers this feature, it is probably with some disappointment that they find support for type selection varies widely amongst third-party applications. Every application that uses StandardFile and Navigation Services dialogs gets type selection for free in those portions of their program. For custom application dialogs and views, developers have been required to write their own, custom type selection code. The result is that some applications don't support type selection at all, some support it to varying extents, and almost none support it in a way that mimics every last nuance of Apple's own type selection algorithm. The opportunity

to end these inconsistencies is before us, for Apple's Type Selection APIs have made their public debut.

Without fanfare, Apple included their time-tested internal Type Selection APIs in the first public releases of the CarbonLib extension for Mac OS. As of this writing, the latest version is CarbonLib 1.0.4, and it is available from Apple's web site at <http://asu.info.apple.com/swupdates.nsf/artnum/n11673>. CarbonLib is the library that enables applications ported to Mac OS X's Carbon APIs to continue running on Mac OS 9 and earlier (as early as 8.1) systems. There is no shortage of compelling reasons to port your application to Carbon. Apple's decision to support two platforms with a nearly identical API set provides an irresistibly easy way of adding Mac OS X to the list of your supported platforms, without losing your existing customer base. Access to standardized type selection is by no means the greatest benefit of porting to Carbon, but it has the potential of causing a vast improvement in the user's experience on the Mac.

WHAT'S THE BIG DEAL?

You might be wondering why something as seemingly straightforward as selecting list items with the keyboard should require a standardized API. The user hits a key, you select a matching item, and we're done – right? Actually, the apparent simplicity of type selection is a testament to its internal complexity. You may not have even considered much of the functionality Apple's TypeSelect APIs provide unless you have spent a great deal of time focusing only on the dynamics of type selection. Most developers do not have the time to become type selection experts, so they implement something that makes sense to them, yet lacks the subtle elegance of Apple's code. Some of the features of Apple's API are:

Daniel Jalkut is a software engineer in the Mac OS X Carbon API group at Apple Computer. In his spare time, Daniel works on his guitar playing and book reading. You can contact him at jalkut@red-sweater.com, or view his home page at www.red-sweater.com.

// Like most Mac developers,
// I easily spend 12 hours a day

// staring at line after line of C++ code in tiny, 9 point Monaco.
// Sometimes it makes my eyes feel like they're on fire.

```
{  
    So the last thing I need is some  
    fuzzy monitor that adds to my headaches.  
}
```

/***** begin excitement *****/

// That's why I'm so jazzed about this SGI monitor.

// Its ultra-high resolution = 1600 x 1024 and dpi = 110,
// giving me razor-sharp contrast.
// And the high refresh rate is
// perfect for poring through lines of code.

// At first, I was amazed at the clarity, the fine details that emerged.

```
{  
    It was like seeing things for the first time.  
}
```

// Later, though, I learned to appreciate the wide aspect ratio [= 16:10],
// with a generous 17.3 inches of viewing area. SGI's 1600SW
// lets me have all my documents viewable at once, and it's
// a flat panel so it fits on my desk with room to spare.

// From the moment I saw this thing I was hooked. You will be, too.

```
{  
    Especially when you find out  
    how affordable it is.  
}
```

// **Check it out.**

/***** end excitement *****/



Michael Whittingham
V.P., Engineering
Wired, Inc.

www.sgi.com/flatpanel

This advertisement was written by an actual engineer. This is the first time in 7 months that he's been seen during the day.
© 2000 Silicon Graphics, Inc. All rights reserved. SGI 1600SW and SGI are registered trademarks of Silicon Graphics, Inc.
Mac is a registered trademark of Apple Computer, Inc. For more information, visit our website or call 1-888-SGI-7373.



sgiTM

International Support

Support for international scripts, which Apple has always promoted, is becoming more and more important as the customer base for Apple outside of the United States continues to grow. As a developer, anything that provides international support for free is a big win for your product, because it is one fewer thing you need to worry about when pushing to make a localized release available. Apple's type selection APIs use script-aware string comparison functions, which in turn guarantee that every selection made by the user causes a match as expected for the appropriate script.

Multi-Character Matching

A major shortcoming of some third-party implementations of type selection is that only the last key pressed is ever used as a criterion for matching against items. If you type characters in a Finder view, you will notice that the active selection changes as the characters you type grow into a more specific match with the item you're seeking. For instance, if there are three icons in a view, "Belize", "Biafra", and "Burundi", typing just "B" will select the first, "Bi" the second, and "Bu" the third. In an implementation that didn't support multi-character matching, the user would be forced to type "B" to locate the first item, and then navigate with arrow keys to the desired item. No amount of typing would select either "Biafra," or "Burundi" without moving a hand from the keyboard to the mouse.

Time-Out and Cancellation

One unexpectedly complicated aspect of a proper type selection implementation is elegantly guessing the train of thought a user has embarked on: knowing when to stop one selection and begin another. This task is impossible to do correctly every time, but Apple comes close with comfortable and consistent behavior that gives the user a great deal of control.

The type selection APIs keep a running tab of the keys that have been pressed, and when no keys have been pressed for a duration of time, that buffer is flushed, with the assumption that the user has finished typing the fragment they were seeking. To accommodate the user who knows they have mistyped, or who has quickly changed their mind, the APIs also respond to the escape key, which forces type selection to clear its buffers and begin matching keys as a new string.

IMPLEMENTING TYPE SELECTION IN A LIST

To demonstrate the use of the type selection APIs, I have included with this article a sample application,

"ListSample", which simply displays a list of selectable items in a dialog. The application responds to keyDown events by passing them to the Type Selection API, "TypeSelectNewKey", which determines whether the key pressed justifies searching for a new match. If it does, another API, "TypeSelectFindItem" is called, which uses the state of the key buffer and a list of items provided by the client to determine the most appropriate selection.

Application Prerequisites

Before I could implement type selection in ListSample, I first had to ensure that the application had been carbonized and compiled with Apple's latest Universal Headers. If you have already carbonized your application, then you're ready to go. If you haven't, then you will need to do so before you can call any of the Type Selection APIs described in this article.

ListSample Implementation

The easiest way to demonstrate the use of the Type Selection APIs is to explain the three pieces of code in ListSample which interact with the APIs:

- A TypeSelectRecord is initialized before any type selection can occur.
- Key down events are passed to Type Selection, and a new match is found if appropriate.
- A callback routine for finding matches is implemented.

First, initialize a TypeSelectRecord. This is done by calling TypeSelectClear() with an empty TypeSelectRecord as its parameter. In ListSample, this is done in main()

Listing 1: main

```

Main
Setup the menu bar, initialize TypeSelection, create our sample
dialog and run the event loop until we are quit.

main()
{
    // Setup the menu bar
    SetMenuBar(GetNewMBar(rAppMenuBarID));
    DrawMenuBar();

    // Initialize the type selection record
    TypeSelectClear(&gTypeSelectState);

    // Prepare the sample dialog
    gTheDialog = SetupSampleDialog();
    if (gTheDialog != NULL)
    {
        // Handle Events!
        while (gQuitApplication == false)
        {
            ApplicationEventLoop();
        }
        DisposeDialog(gTheDialog);
    }

    ExitToShell();
    return 0;
}
```




Some companies are really on the ball.



Some of the world's most forward-thinking companies have transformed their Web sites into live meeting rooms. They have discovered the speed, reliability and scalability of the WebEx Interactive Network. Their

Web sites now hum with people conducting business right in their browsers, without any hardware or software changes. Now you can give presentations. Share software and desktops. Tour the Web. Voice and video conference.

All in real-time. All on the Web. Talk about ROI. Isn't it time your company got on the ball? Visit webex.com or call 1-877-50-WebEx to see how great minds meet online.

great minds meet online at webex.com



Next, for each keyDown event that pertains to the list, ask if a new item needs to be matched. This is done by calling `TypeSelectNewKey()` – if it returns true, then it is time to find a new match for the list by calling `TypeSelectFindItem`. I have implemented this functionality in the `ListSample` routine that handles all keyDown events:

Listing 2: HandleKeyDownEvent

HandleKeyDownEvent

Handle key down events not handled by the Dialog Manager. If the key is not a menu-shortcut, then ask the `TypeSelection` APIs whether the key might change the state of the current selection, and if so, ask it to determine the new selection

```
void HandleKeyDownEvent(EventRecord* theEvent)
{
    Boolean          eventHandled = false;
    static IndexToStringUPP myStringGetter = NULL;

    // Handle the cmd-key menu case first
    if(theEvent->modifiers & cmdKey)
    {
        long menuKeyResult;

        menuKeyResult = MenuKey(theEvent->message &charCodeMask);
        DoMenuSelection(HiWord(menuKeyResult),
                        LoWord(menuKeyResult));
    }
    else if (gTheDialogList != NULL)
    {
        // Only allocate the IndexToStringUPP once
        if (myStringGetter == NULL)
        {
            // This call-back function is used by Type Selection to
            // fetch elements of the list the user is navigating.
            myStringGetter =
                NewIndexToStringUPP(GetStringFromIndex);
        }

        // Ask Type Select APIs whether we need to re-check
        // the selection
        if (TypeSelectNewKey(theEvent, &gTypeSelectState))
        {
            short    listCount;
            short    newListIndex = 0;
            Cell      newSelection;

            // How many items are we dealing with?
            listCount = (**gTheDialogList).dataBounds.bottom;

            // Ask Type Select for a new index, based
            // on the current state of typing
            newListIndex = TypeSelectFindItem(&gTypeSelectState,
                                             listCount, tsNormalSelectMode,
                                             myStringGetter, NULL);

            // Unset any selected items before choosing
            // a new selection
            SetPt(&newSelection, 0, 0);

            // Starting at the beginning, get a selected cell
            while (LGetSelect(true, &newSelection, gTheDialogList))
            {
                // Unselect it
                LSetSelect(false, newSelection, gTheDialogList);
            }

            // Set the new item to selected
            SetPt(&newSelection, 0, newListIndex - 1);
            LSetSelect(true, newSelection, gTheDialogList);
        }
    }
}
```

```
// Auto scroll to make sure the selection is visible
LAutoScroll(gTheDialogList);

// Workaround to List Box control behavior - LsetSelect
// causes an immediate redraw into the list's port,
// which in the ListBox control case is an offscreen
// buffer allocated by the control manager, and doesn't
// cause an update event to be generated for the parent
// window, so we need to force a redraw.
DrawOneDialogItem(gTheDialog, rDialogListBoxItem);
}
```

One of the parameters to the `TypeSelectFindItem` call is a callback routine that you provide to allow `TypeSelection` to determine the elements of the list it is choosing from. Depending on the circumstances of your implementation, the items you are selecting from might not be a straightforward list. In our case the routine is quite simple. It simply looks up the desired index in the list control we are searching in:

Listing 3: GetStringFromIndex

GetStringFromIndex

Callback routine for the type selection routine `TypeSelectFindItem()`. This routine is called to fetch the items in a list, and determines which of the items is the best choice considering the key strokes that have been pressed.

```
pascal Boolean
GetStringFromIndex(short theIndex, ScriptCode *whichScript,
                   StringPtr *whichString, void *ignored)
{
    #pragma unused (ignored)
    Boolean    returnValue;
    static Str255 thisString; // static because we return a
                             // pointer to this string

    // Set the script - in this sample, we know that
    // all the list items are in Roman script.
    *whichScript = smRoman;

    if (gTheDialogList != NULL)
    {
        Cell desiredCell;
        short stringLength = 255;

        // Fetch the item from the list,
        // and get the cell data (a string) into the result

        SetPt(&desiredCell, 0, theIndex - 1);
        LGetCell(thisString + 1, &stringLength, desiredCell,
                gTheDialogList);
        thisString[0] = stringLength;
        *whichString = thisString;
        returnValue = true;
    }
    else
    {
        *whichString = NULL;
        returnValue = false;
    }

    return returnValue;
}
```




Sanctuary.

Move to **digital.forest**. The leading Internet Service Provider for the Macintosh community.

You need maximum flexibility, security and performance from your ISP. You'll find it with digital.forest. We're the proven provider of Macintosh Internet service.

Founded in 1994, our specialty is Mac systems and technologies. In fact, digital.forest is the world's largest Macintosh Hosting Provider. And our hard-earned knowledge of Mac server software is the broadest in the business.

We pioneered Macintosh Server Colocation and FileMaker Pro database hosting services. And our commitment to service keeps growing. So whether you choose us to host your files or house your servers, you've chosen the best.

Go with the industry leader. Discover the sanctuary of digital.forest.

digital.forest



8 7 7 - 7 2 0 - 0 4 8 3

info@forest.net
www.forest.net

Call toll-free in the U.S. and Canada.
For international inquiries, please call 425-483-0483.



FileMaker Pro is a registered trademark of FileMaker, Inc. The digital.forest logo is trademark or registered trademark of digital.forest, Inc.

SUMMARY

Type selection is certainly not the top selling point of Carbon. The APIs I have described would have been appreciated if they were publicized years ago in the classic Mac OS arena, but at least they have finally arrived in Carbon. I hope this article has demonstrated that it's not difficult at all to implement this functionality, and that the benefits to the customer are overwhelming. I hope to see all of your applications sporting fancy new type selection capabilities in their next releases. Happy typing!

ACKNOWLEDGEMENTS

Thanks to Darren Litzinger for reviewing this article.

*Want to have the entire history of
MacTech Magazine at your fingertips?
Check out the MacTech CD-ROM with
superfast searching.*

*For more info, see
www.mactech.com*

*or write
custservice@mactech.com*

The type selection APIs consist of only four routines and one callback routine. Here is a short description of their functionality:

TypeSelectClear

Used to initialize a TypeSelectRecord, which is the data structure that holds the state of type selection at any given time. Call this routine at least once, when your application is starting up. After you launch, only call this routine if you want to intentionally void the typing the user has made at a given point.

```
void TypeSelectClear(TypeSelectRecord *tsr);
```

tsr Points to a TypeSelectRecord, which requires initialization.

TypeSelectNewKey

Every time your application receives a keyDown event that might pertain to selection of a list item, you should pass the event to this routine. It examines the current buffer of characters and the value of the key event it is receiving, and returns true if the new keystroke has warranted the need to update the active list selection.

```
Boolean TypeSelectNewKey(const EventRecord *theEvent,  
                          TypeSelectRecord *tsr);
```

theEvent A pointer to an event record containing a keyDown event.

tsr Points to a TypeSelectRecord previously initialized with TypeSelectClear.

TypeSelectFindItem

When TypeSelectNewKey returns true, use this routine to actually determine the most appropriate list item to receive the new selection. This routine takes as a parameter a callback function that you use to supply the algorithm with the contents of your list.

```
short TypeSelectFindItem(const TypeSelectRecord *tsr,  
                         short listSize, TSCode selectMode,  
                         IndexToStringUPP getStringProc,  
                         void *yourDataPtr);
```

tsr Points to a TypeSelectRecord previously initialized with TypeSelectClear.

listSize The number of items in the list you are selecting from. If the number of items is unknown, pass 0xFFFF; and be sure that your callback function returns false when a requested index is not found.

selectMode Specify one of "tsPreviousSelectMode", "tsNormalSelectMode", or "tsNextSelectMode" to request the item before the matched item, the matched item itself, or the item after the matched item. The previous and next modes are what the Finder uses to respond to the Tab and Cmd-Tab keys. Typically you will pass tsNormalSelectMode.

getStringProc Pass a UPP to a routine that will fetch strings by index from the list of items you are selecting from.

yourDataPtr Pass any value you would like to have passed back to your callback function.

TypeSelectCompare

TypeSelectCompare is used to compare specific items from a list, using the exact same comparison that TypeSelect would use in a call to TypeSelectFindItem. This is useful if you have a sorted list, and want to optimize item selection based on knowledge about the sorting in the list. TypeSelectFindItem compares each and every item in a list, so for very long lists pre-sorting and using TypeSelectCompare might result in a performance gain.

```
short TypeSelectCompare(const TypeSelectRecord * tsr,  
                        ScriptCode testStringScript,  
                        StringPtr testStringPtr);
```

tsr Points to a TypeSelectRecord previously initialized with TypeSelectClear.

testStringScript Indicates the script of the string that is being compared.

testStringPtr Points to the string that is being compared.

MyIndexToStringProc

This routine is defined by your code, and serves as an access point for TypeSelection to determine the items of the list that you are selecting from. When called, you will need to fetch and return a string from your list that is indexed by the given item number. You must return both a pointer to the string and the script code for that item.

```
Boolean MyIndexToStringProc(short item,  
                            ScriptCode *itemsScript,  
                            StringPtr *itemsStringPtr,  
                            void *yourDataPtr);
```

item The index of the list item being requested.

itemsScript You return the script of the requested string here.

itemsStringPtr You return a pointer to the requested string here.

yourDataPtr A reference pointer for whatever you choose.

development revolution.

You're part of the movement.

At the core you'll find two innovative technologies allowing you to deliver the most comprehensive software solutions available - 4D and WebSTAR.

Time is short. Seize the moment.

Maximize your efforts using revolutionary development tools and Internet Servers that deliver now and into the future.

Don't waste today.

Download your future. Ignite your potential.



W E B S T A R . 4 D . C O M
1.800.881.3466

By David K. Every

WebObjects Development Lifecycle

What does WebObjects do?

ARTICLE SCOPE

There are many parts to making a successful web-solution. The success or failure will be based on how good you (or your company) is at being an engineering company, engineering team or at least engineering project. If you aren't an entire engineering company then you are going to have to emulate all the facets of being one. If you don't have the size and experience, and are approaching the problem on your own or with a small team, then you need to try to all wear many hats in order to make sure all the parts of the project run smoothly. Each part is important and will help you avoid pitfalls — so don't discount them.

It is beyond the scope of this article to explain all the concepts necessary to manage a successful team, how to do proper analysis, scoping, design, project management, as well as explaining how to be an interface designer, DBA and data architect. But it can sum up some of my experiences (many hard learned), and hopefully give the reader an introduction or review of the high level concepts on what to do, what not to do, and what are the warning signs to look out for.

This article is also tailored towards WebObjects project management — assuming an “average” project. Your project might not be average — but you can think about how you can borrow ideas and concepts, and how this information might help you..

BALANCE

The key to creating a successful project is balance. Always remember the balances and tradeoffs. You have to balance your goals, with the customers desires. You have to balance marketing with engineering and with financing. You have to balance features (power) against simplicity (learnability) — and form (style) against function (usability). You need to know your customers (marketing and feature), but you need to know what you can develop (engineering) — all balanced with what you can afford in time and money. There are many seemingly conflicting goals, and all of them have to weighed against the others and balanced. Everyone is not going to get exactly what they want — but if each gets to keep the important things, and they get it in balance with the others, then the product will probably be better than if there is no balances at all.

Balancing the needs (and communicating them) is done in order to get everyone on the same page, and shooting for the same goal. Even if meetings seem useless or time wasting, they are valuable if you can build some consensus or let people feel like their views have been aired (and really listened to). Even if they don't get what they want, you are more likely to get buy-in if they feel they have a voice and win some of the time. Even spending time exploring paths you aren't going to follow can be very important, if they can at least explain why the company isn't going to do something, or why things are being done the way they are. This makes your decisions more defensible and better understood. So meetings can be a tool to create consensus and understanding or they can decay into adversarial places where there is little real communication — always strive for the former.

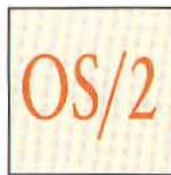
Conflict is going to be natural in the process — people are each advocating different views or different parts of a solution which are critical to them. Accept that conflict, don't take any of it personally, and don't hold grudges and try not to allow others to. You need to express a view strongly, and get over the fact that others won't agree and you aren't going to get everything that you want — if you can't do that, then

David K. Every is a Senior Software Engineer who has done many enterprise and client-server (n-tiered) solutions, a WebObjects programmer, and the creator of the MacKiDo website (<http://www.MacKiDo.com>). You can reach him at dke@mackido.com.

The *True* BASIC[®] story
is a *powerful* ~~simple~~ story.

Write your code once.

Run it (without change)



here, here, here, here, here, here & here!

Demos and information from our web site:

<http://www.truebasic.com>

TRUE BASIC INC • Founded by the *Inventors* of BASIC

PO BOX 5428 • WEST LEBANON NH 03784-5428 • 800 436-2111 • 802 296-2711

you certainly shouldn't be involved in engineering. Remember that you all have the same goals — to build the best product possible, in the least amount of time (with the lowest costs). Engineering is about tradeoffs and balancing all the differing desires — but you are all shooting for the same prize. Keep reminding people of that, and keep reminding them that things will evolve and not getting something now, may only mean for a while. Good ideas will keep coming back, and eventually win — and often things lose because it is just not yet the right time for that goal, but not that their goal itself is bad.

Unfortunately there is no magic to knowing where the lines should be drawn, when a project is unbalanced, and what to do about every situation. Nor is it easy to know where to cut off communications on certain paths, or all the nuances of managing people and delicate professional egos. That is all stuff that you just have to learn by experience, and there are books more targeted towards those subjects. However, if you at least remember to watch for balances, make sure that people are being heard and that your team is building consensus (and not hostility), people are learning why and where compromises are being made, and you are keeping communication open and constructive (with the eye on the prize), then at least you have a head-start on creating a successful project.

MANAGING THE PROCESS

There are many books on the theory of analysis, design, and development. Many explain how to decompose problems from different points of view, and different methodologies for OOA (Object-Oriented Analysis) or OOD (Object-Oriented Design) — and many more on the older procedural analysis and design processes. Some explain techniques for drawing how things relate, nomenclatures, methodologies, and mental exercise to break problems down, and so on. This article is not one of those books. I've read many of those books, and I'm not knocking them — they have value and I recommend reading many of them, they have many nice insights that can help. However, most companies just don't have a staff of people all trained in the same methodology, at all levels of the company, in order to do a good job with them. Without a deep understanding of the processes (at the management as well as at the engineering level) then the processes can quickly become less than efficient — and the results will probably not meet expectations.

The point of what analysis and design is really about should be as much about getting people on the same page, getting them communicating well, managing expectations and getting some consensus and agreement on what will be done. It is setting targets that people understand and can shoot for, and measure progress against those targets so you can get better over time. The processes, documentation and drawings, and exercises in many analysis books and classes are not the ends — they are just ways to try to achieve the ends (communications, consensus, repeatability in forecasting, setting goals and managing expectations).

Many of the larger, complex processes may be necessary and valuable on larger projects (say 15 engineers or more) with multiple teams, huge development cycles (over a year), and huge amounts of investment. However, successes in those processes are far more rare than the failures — especially if people aren't willing to be adaptive, or willing to accept the facts on scheduling or scope that is often staring them in the face. I've found that those processes are usually overkill for creating most web-applications.

WebObjects is a powerful rapid application development tool that makes programmers much more efficient than many other tools. This magnification of programmers efficiency means that smaller teams can do more — and thus more projects are smaller with WebObjects than they would be without it. So the project management processes (including analysis and design) can be a little bit lighter as well.

If you have a large team, of highly trained people, or a well established process that is working for you, then I'm certainly not suggesting that you throw it all away and try to mimic the suggestions of this article. Stick with what works. But there are many who are unsure about how to approach WebObjects development, or looking for alternative ways of thinking about that problem. This article is about techniques that I've learned and used in startups and larger companies to help establish a sensible process (or streamline one) for doing WebObjects development and project management. These are ideas for how to manage a WebObjects project (smaller, rapid development projects), while still maintaining a useful amount of documentation and process in order get things done right, and to still allow engineering enough critical communication to work well with other groups such as Quality Assurance, Documentation, Marketing, Management and so on.

GO VISUAL

If you verbally describe something to a group of people they will each imagine something different — they may think that they are all talking about the same thing, but in their minds they are picturing something that might be quite different from the rest. When one person delivers on what they were saying, the others often find that it didn't meet with their expectations — and nothing seems to get people more frustrated than having something fail to meet their expectations (ask any married man). The further away you are from their expectations, the worse things are likely to be — so one of the keys to successful project management, and a happy life, is good communications and managing expectations.

If you write a document that describes how something works, people are more likely to be on the same page than just speaking the same thing. Speech is more imaginative than writing, and words on a page can be less ambiguous. Also people can digest written documents at their own speed, and review until they get it. So a well written set of design documents can really help facilitate communication.

Unfortunately, people are still victims of their ability to write, read and digest these documents — and those are skills that take time to master. Most design documents are not very good and tend to take up lots of time to create, with little results, and tend not to survive confrontation with implementation — so by the time the product ships they are woefully out of date. Many people aren't willing to read long design documents — and just ignore them. So while design documents are usually better than just verbal concepts, written words usually aren't good enough to get people to understand what others mean.

Most people tend to be visual and understand what they see, and they don't have to conceptualize the intangible based on your description. Thus the key to successful documentation (or many discussion) is pictures and drawings. This is why many restaurant menus have pictures — don't describe it, just show it. Don't be afraid of a white board, a sketchpad, or a drawing program. It is cliché, but sometimes a picture really is worth a thousand words — and people are more likely to use the picture than they are to read the thousand words. A few simple drawing, with notes and points can replace dozens of pages of wordy documentation. Because of this, I find to be the best form of design document, and tool for project management is the prototype!

Prototyping is the quickest way to get people to really envision the goals. And the quickest prototypes I've ever used are just a series drawings, with a few notes to describe what each thing does, and where items are labeled to go to other drawings. At the first phase, I used to do this with a bunch of printed pages, and page numbers for each link, button or image, and just through the stack to show people where each was going. Later, I automate the navigation with some prototyping tool (like HyperCard or VisualBasic). With HTML and websites, it is even easier to just link a bunch of static pages together and use that as a prototype that people can see and feel and sign-off on.

With all this in mind, here is my simple process for analysis, design, documentation and project management.

ANALYSIS

- 1) Analysis starts with information collection. Collect all the features and ideas that people want, and build a list (so many ideas won't be lost). And this is just the features and ideas people want for the first public release — keep a separate list for the far future stuff (that list will be revisited after you've shipped 1.0).
- 2) Prioritize the 1.0 list, and divide it into three or four groups. Those groups will be iterations of releases. (I'm a big fan of iterative development — or at least a hater of waterfall development, since I've never seen the latter actually meet expectations, and it always causes undue stress and frustration). The first few iterations are your targets for the first release — the last few are your visions for the future or "nice to have's".



Yellow Dog Linux™
Champion Server 1.2.1

\$25-\$100 packages
Pre-Installed drives

Over 1,000 applications
Bootable Install & Rescue CDs
All source RPMs on 3rd CD

Enjoy Mac-on-Linux, AbiWord,
Netscape, Apache, 3 databases,
beautiful GUIs, and "yup!" our
automated FTP update utility.

Professional. Powerful. Proven.



www.yellowdoglinux.com

- 3) Most of the effort should be focused on things that will make the cut. You can, and should, capture a few good ideas that can be in the other passes — but focus on getting people through the first iteration of the cycle. Remember to layer functionality — get the basics working, then add the “extras” later on. Let people make their mistakes early in the process when they are less costly — let them learn what to do and what not to do before you are under the high-pressure end-game. The key to the early passes is to learn.
- 4) Now go on to the next step and use the design process to mature your analysis — you don't have to do all the analysis in the first stage of the first iteration of the process, let the next step mature the previous one. The goals are to get most done and revisit over time — not get everything done at once. Feature creep, design creep, and code-creep happens. Don't fight it — just plan for it. Like a fine wine, let things mature on their own time.

The steps sound simple — but it can take quite some time just to get through that. In order to keep on track, try to keep the analysis from getting into specifics and details — it is easier to just manage the higher level feature list, without letting people war over the details of implementation. Don't try to do it all in the first pass. Make sure people realize that there are other iterations in the process — and that things will get in and that later planning will get into details.

No analysis is perfect. In fact, I've rarely ever seen analysis that was really close to the final product — and many of those that were, turned out to be exactly what was asked for, just not what people wanted. No one is omniscient, and since you can't anticipate everything up front, things are going to have to change. So you have to be adaptive, don't try to capture everything — just try to capture what you have at each step. Keep coming back, and keep things up to date — but a series of small meetings and capturing what you have gives people a sense of accomplishment. Then come back and keep making those baby steps. If you try to do it all in one series of meetings, everyone will end up frustrated or bored, and you'll miss major chunks of functionality and lots of good ideas. If you keep revisiting over time, and capturing a few things on each pass, people get renewed and refreshed to add more each time.


Tip: It is better if you don't go into meetings empty handed. It will dramatically reduce the time to develop if someone is in charge of being the architect, or chief interface designer, leader, and they come into the meetings with a rough draft already sketched out. That person can capture half of the information via one on one meetings with the others before the bigger collective meetings — this is a big head start. When a meeting starts, it should be a continuation of where people left off — and just be about capturing any extra stuff that what was left out. Collaboration is useful but time consuming — so if you can get all the common stuff out of the way, then you are only left with the important part of

USE CASES

There is a school of analysis that believes in using “use cases” to do the analysis. A use-case is a path through operation of a system (or the flow of one feature). Usually one per role and function. Just follow a feature-flow to do one thing the user might want to do. Then add in others. When you document enough of the various paths (work flows), then you capture the behaviors necessary to complete the design.

All this is pretty good theory — and use cases aren't a bad way to do some of the analysis. However, I've yet to see a set of use-cases that were complete, or any analysis that was ever complete for that matter. If you try to do all the analysis up front — you'll come slamming into a wall near the end of the cycle, as everyone starts to figure out what was missed. Or else people spend far too much of the development cycle trying to figure everything out up front — when that knowledge would have been obvious (and cheap) if they had just been willing to wait for the omission to come to them. And you want to get everyone moving in the process in small steps (baby steps).

Analysis without design and development time to help mature it, is like trying to create a recipe for great food without touching any ingredients, or an oven. It can be done, in theory — but in practice, it never seems to work well. So start high level (very high level), then get an iteration done, then add more detail and functions and features in the next pass, and so on. Make a skeleton product (analysis, design, implementation, quality assurance, documentation), then repeat the process when you have a better idea of scope, time, and what everything is going to take. This will help you make your big mistakes early (when things are small) — so that you can recover easily and get back on track. The longer you wait to complete every step of the process, the later in development you'll be when you discover the flaws — and the more time/energy and money it will take to correct things.

Analysis, and use cases are valuable exercises that produce results and can easily save far more time than they cost — if you don't abuse them. Use-cases in particular have always been a useful alternative way to help exercise the design and other analysis, and to give some groups like QA (Quality Assurance) a head start on building their test plans. But I've found that use cases compliment the normal analysis, and aren't really a replacement of it — just a more detailed way, or a different way of doing it. 

collaboration (which is the areas that need to be discussed, and not all the stuff where people are in fierce agreement). This head start will get meetings completed in a fraction the time — and a good moderator (who keeps things on track, and schedules follow up issues for other meetings) will help get meetings over in a fraction the time as well. This makes

meetings more useful, productive and satisfying.

Another important meeting tip is the duration. 1 hour meetings are about optimum (unless you can do them in less time). After a couple hours, people completely glaze over and start nodding to anything just to end the meetings. This is the EGADs factor - "Everyone just doesn't Give-A-Damn" when a meeting is running too long — they are just thinking of escape. The only thing worse than this, is the personalities that may become so pissed off (they're tired and frustrated), that the meetings turn hostile, and the meetings turn into a slug-fest. So it helps in analysis and design meetings (or all meetings for that matter), to break them into many smaller meetings (one or two per day), than to try to rush it and cram it all in at once. Let the process happen. If people start getting tired and cranky or apathetic, it is time to break it up and come back on a day when you'll be more productive. And sometimes if people are tired of one subject, it is better to just skip it, and address the subjects you can agree upon, and try to come back to "sensitive issues" later (another day or week).

HIGH LEVEL DESIGN

Once you have a working set of high level requirements (which is what the analysis should have given you'll), use the design process to mature the requirements and create the lower level requirement. (High level design and low-level requirements are almost the same thing).

For a website, the high level design starts at the navigation, and then putting what function is at each point in the navigation. That is it.

ANALYSIS VERSUS DESIGN

It can be difficult to figure out where analysis ends (or should end) and where design begins. In larger organizations they try to divide up tasks so that they can throw more bodies at a process — but sometimes this is like 9 Women trying to rush and have a baby in just one month, it doesn't work well.

Analysts job are to do analysis. Because they specialize, they can add value to the process and offload the team. But there are also programmer-analysts who need to do to low-level analysis (and design). When analysts try to do the design, or you break up the analysis and design processes too much, then the analysis is abstracted from how the task will be implemented — and this usually costs in development time. It may allow more junior programmers to do the implementation (assuming the analysis and design is perfect — which it never is and thus parts of the work have to be done over again, and again), or it frustrates senior engineers by treating them like more junior ones (and this can lead to conflict between engineers and analysts). So to do analysis and design effectively, you need to keep the analysts and the programmers (and designers) working together, not separate. Each has value to contribute to the process, as long as they are kept in balance.

MT

Build a hierarchy (outline) of the site map, and how a user will navigate.

On larger sites there may want to break the site into a few (or many) smaller sites. Each small site will be around a branch of functionality, or a role.

Add in a list of functions for what each stop on the hierarchy should have on it (what features and functions are available).

Other issues and ideas will be brought up that will need to be added to the requirements, or at least discussed as requirements. Don't try to solve them all at this level — maybe adding one or two of the obvious ones won't slow you down, but collect a list of any of the more ambiguous ones, or ones that are going to cause contention to bring back to the analysis review meeting. This will keep the design meeting moving — solve the problems you can solve, and skip the ones you can't. And this will also allow the design to mature the analysis.

At the end of this, you should have a site map — which is a nice visual way for you to see how people will navigate, and where they will get to various features and functions. This visual representation of where things go will help things mature much faster than it would have as just a spec. And you will be able to see that all the functions are captured and available somewhere.



Web Server 4D 3.2
<http://www.mdg.com>

WS4D/eCommerce
a single application
that does it all !

Web Server
Database Publisher
eCommerce Server
Handles Virtual Domains
and all your Databases

Hosting Services Now Available

- **WS4D & WS4D/eCommerce Hosting**
- **Co-Locating (including 4D & 4D Server)**
- **We are 4D & 4D Server experts and have been working with 4D since 1988.**

Database Hosting \$50/month
eCommerce \$100/month
Co-Locating \$400/month

To find out more about MDG Hosting, visit:

<http://mdg.com/hosting.html>

Tip: I tend to use the web to design the web — since I believe the purpose of the web is communications, this is “eating your own dog food”. I make an intranet site for development that contains all the notes/documentation from each step of the process. That site would contain (at this point) a simple high level requirements document, and a static site (navigation) map with each page being a list of functions that page will do (and maybe what it needs to contain). You can do this on paper — just the website is more interactive and publicly available, is easier to update, and saves trees. The designers and principals can now see what is where, and truly feel the hierarchy and the beginnings of a product. Everyone will start understanding the scope of the product and feel progress.

Low Level Design

Get an interface designer and graphic artist to create the first pass look. I call this the style-guide — what is the style and look of the site. This is just a simple picture for what each types of page will look like. Get a first pass of this done sooner because it gives people a lot of satisfaction to see some of the goals they are shooting for. Avoid needless battles over details, remember to remind people that this is just a very rough first cut (to get started) — and the end result may not look anything like this.

Decompose the common elements — both visually and functionally. I often do this with a User Interface Guidelines document that has a list of what rules that I’m going to try follow. Know that this too will mature over time — so it starts small and loose, and will fill in as you go. Create a list of common features and functions that are repeated on many pages — and try to put these on all (or almost all) pages, and in standard places. This will allow reuse of code, and create an interface metaphor (set of common behaviors that users will learn).

Now revisit each stop on the site-map — and instead of having just the feature list on each page, create a simple sketch of where things will be. Since you have set of UI rules for how things will behave, and a first pass at what things will look like, these sketches only really need to capture placement since you already have more details about look captured in the graphical style document, and the details of the feel is captured in the UI document. I usually use rectangles with what is going to go where, though some people want more detail and like to actually put in place detailed picture from the UI guidelines and Graphical look to make a detailed mockup.

Again, this phase of the process will mature the first two (analysis and high level design). The low level design will probably require changes and maturing of the high level design, and to a lesser extent modify the requirements. So capture any issues that are brought up, and quickly get past the ones you can’t answer. Do what you can. Let things happen in their own time.

At the end of this, you not only have a site map, but you should be able to walk the pages with not only a list of what

TOP-DOWN OR BOTTOM-UP

Some people design the back-end first, or do the data-analysis first (bottom-up design). I tend to find that top-down (interface to model) is much better because it is less abstract and more visual.

Most non-engineers don’t understand ERD’s or data-dictionaries or data-maps (the stuff you do at the bottom or from the data point of view). The non-geeks see these things and just just nod, and say, “OK”, or “NO” without understanding what they are agreeing or disagreeing with. When you finally get around to implementing something, the higher-ups are less likely to say, “that’s not what I thought you meant” when they’ve seen it visually (on screen shots and mockups). Without an interface and that buy-in, it is impossible for a database to take into account all the requirements of the interface and application. So in bottom-up design, changes won’t happen until you get to the top (interface) — and you want to get as many changes out of those the way as soon as possible so they have less impact, and so people understand what they are getting.

Once you have the tangible and visible map of how the product will work (and look) people have much more faith in what you are doing — and you can show them exactly why you need a data element in the database or model, and what it is all used for. And you can be sure that you are designing for the right target!

Remember, you are building a Web Application. The applications features and functions define what must be in the database — not the other way around. If you start with the database and data model (in design), what you are doing is building a legacy database that you will then have to support with an application — instead of building an application that has a database supporting it! The top matures the bottom a lot better than the bottom matures the top.

Nothing is black and white. I’m sure there are a few cases where you have huge data requirements, and almost no front end or interface requirements — for those, I would do more a bottom up design. But if you are doing that, then you aren’t really building a web application — your doing a database with a web front end. Heck, if you are doing that then you are building just a database with a thin presentation layer, WebObjects offers a “direct to web” function (or direct to Java function), to enable you to just map the data to a simple user interface (if that is all that you really want to do). So by definition if you need to manage a project in WebObjects, you are probably doing a Web App and you should be thinking Application centric, which means top-down thinking (and not bottom up).

For the most part, you actually do a little of both (top and bottom design) at the same time — and you actually start collecting the data from the interface as you go, and you can be working on the data set and data model in parallel. Just try not to lead with the database design. Sometimes there is no choice — and you must learn how to make an Application to work with a legacy database — but it never as easy as having a clean slate or going the other way.

MT

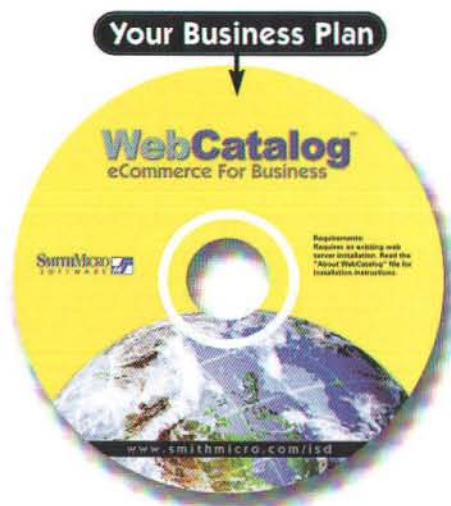


**Put eCommerce to work for you and
change your business model.**

WebCatalog is a complete eCommerce and dynamic web publishing solution. Utilizing existing web server software, WebCatalog provides the capability to design and operate online storefronts with all the features found in leading sites. Build unlimited storefronts with our Storebuilder wizard directly from your browser - even upload your product graphics! WebCatalog provides multiple platform support for Windows, Macintosh and UNIX. It's the easiest way to take advantage of the web and to change how...and where... you do business.

- Unlimited storefronts
- High-speed internal database
- WAP-enabled sites
- ODBC, SQL support
- Electronic shopping cart
- Invoice calculations

1.800.477.1543 • www.smithmicro.com/isd



SMITHMICRO
SOFTWARE

©2000 Smith Micro Software, Inc. All rights reserved. All trademarks and registered trademarks are the property of their respective holders.

is going to be on each page, but a little sketch of where things might be placed. You also have a rule book for how you'd like things to behave. (Of course guidelines are not rules — just goals. They will have to change over time as well, and there will probably be exceptions — but the goal is to follow them as much as possible, and any deviations must have good reasons). Lastly you have a feel for what things will look like with the style guide.

Tip: Again, I tend to use the web to design the web. I tend to put the interface guidelines that are created, and the general look of the site on the developers intranet site. Since you already have a map how the site will navigate, with what features it will contain (place holders for each page), now you can just add what it will look like (roughly) on each of those pages (in place). I put in the sketches for each page, on each page in the hierarchy (with some direct navigation links). This allows a simple working mockup, where people can navigate page to page, and see what each page might look like (very roughly) and what will be there. At the bottom of each page is a list of things this page should do, or at least a link to a page that has that information. And you need a section of the site to map what the common elements (that will be reused on many pages) will look like and need to do.

DATA MODEL AND BACK-END

Now that you have collected the basic features and requirements for the interface, the database requirements should all be laid out for you.

- 1) Collect all the data elements that will be needed from each of the screens.
- 2) Pool the data in logical groupings — what belongs with what. How do they relate to other items?
- 3) Normalize the data so that common elements are in their own tables (and don't have to be entered more than once), and group data in a logical manner (often as close to mapping to the screens as is easily possible) and you should have a pretty good back end.

There is a lot to doing a good data map that is beyond the scope of this article — just like there is a lot to doing good graphic arts, or good UI design that is also beyond the scope of this article. But at least these are the steps you can take.

This step in the process will help you find elements that you forgot for other pages (low level design), and can easily ripple up to the high level design and requirements. Go with it. After this stage you should have a system design that people can really shoot for. You have the top, and bottom. You have the basics of a look and a feel. You also have the behaviors of what each screen is going to have to do. You are now ready to go.

Tip: I tend to use the web to design the web — since I believe the purpose of the web is communications, this is "eating your own dog food". I make an intranet site for

development that contains all the notes/documentation from each step of the process. That site would contain (at this point) a simple high level requirements document, and a static site (navigation) map with each page being a list of functions that page will do (and maybe what it needs to contain). You can do this on paper — just the website is more interactive and publicly available, is easier to update, and saves trees. The designers and principals can now see what is where, and truly feel the hierarchy and the beginnings of a product. Everyone will start understanding the scope of the product and feel progress.

MIDDLE LAYERS (BUSINESS LOGIC)

The last thing you must do is start figuring out how to hook the back-end to the front end, and the design is done. On many small or mid-sized projects, it will go pretty quickly and there will be little need for a lot in-between (all the mid layers). On larger project, or complex project this can be quite an effort — but all this is traditional, and WebObjects doesn't change how these processes are done.

Often the list of features that are on the site map are the business rules. The what does this page have to do (from a business sense). Once you start trying to implement them, you find out that those high level rules may take a lot of back-end code. Many of these will be your middle layer. If you embed them in either the front or back end (interface or database respectively), then you will usually make it harder to evolve and grow those over time. So often these rules and functions go in their own middle layer.

The other thing that the "middle layer" is for is a place to put commonality and things that you can reuse. Just about anything that is going to have to be used on more than one page, or on more than one entity, is a good candidate for the middle layer. This improves code reuse — which means writing less code, supporting less code, and of course few bugs (since repeated behaviors are implemented in only one place — so one fix, fixes all cases where that bug can manifest).

If you just keep a watch out for things that can be reused, or things that don't really belong in the interface, nor in the entities themselves — then the middle layer will mature itself, and all you should do is just keep a running list of what is in there, and where it is located.

PROJECT MANAGEMENT

To do project management, you need to manage time, resources and scope. You can't do that effectively until you understand scope. Every step in the process was just leading to this point. Now that we know scope, we have a starting point to calculate the other things — but first we have to project rough cost of the scope.

- Custom Destinations

- Electronic Transaction Processing

- Easy Trialware Creation

- Install or Uninstall

Get It Together With InstallerMaker™

- Installation From FTP or HTTP Sites

When Time Is Money, Get It Done Fast!

Build Smaller Installers - With StuffIt InstallerMaker, you'll build installers faster than ever before! Compress your installers an average of 15% smaller using the power of the StuffIt Engine®. Smaller installers download faster, provide added space on CDs and servers, and increase network bandwidth.

Quickly Create Demoware - Easily turn your application into a polished demo. Just set the number of days for the demo to be active, paste in the graphics, and you're done!

Archive Freshening - Automatically update your installer project file; eliminate repeated searches for modified files.

- Resource Installation

- Built-In Resource Compression

- ShrinkWrap Disk Image Support

Scripting Saves Time - InstallerMaker provides full scriptability for all features. Automating the build process saves time and helps ensure the integrity of your installer.

Trialware in Minutes - Creating trialware is a breeze with InstallerMaker. With just a few clicks, and no extra coding, you can create trialware that is e-commerce ready.

- Marketing Opportunities To Help You Make Money

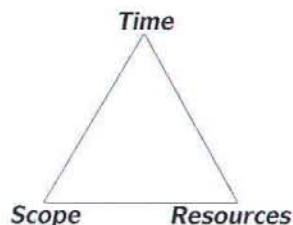
Update in a Flash - Easily build intelligent "diff" files in installers to create small updaters for quick online distribution. From one simple installer, you can update 68k, PowerPC or FAT applications.

- Hierarchical Package Support

More Details Online - There's a lot more InstallerMaker can do for you. Get all the info at www.aladdinsys.com.

- Built-In Updaters





I found the easiest way to do project management is to just walk the site and documentation that I've had to already create, and do time estimates on everything. There is a map of the entire site, with pages that explain what you need to do on each, and you've extracted all the common elements as well. What else do you need? Just put a rough time estimate on each "page" on how long it will take to complete. I usually measure this in days, with 1 day being the minimum, and with a week being a very complex page in WebObjects (that probably should have more of it's elements broken into sub-components). Do the same for the global components — and the interface scope is estimated.

You still have to factor in the costs for the model — but that is usually a smaller scope than the front end. EOF is very fast at building the model, and then mapping the model to EO's (entity objects). So most of the programming load in this area is in common helper functions and business logic (the middle layer). But both the back end, and the front end design should give you a pretty good idea of what is required there. Just write down all the things you need to do, give them a rough load estimate, and add them all up. Those routines can vary far more in time to create depending on your business logic, so I don't know of any hard and fast rules — I just tend to approximate based on fractions of a day, to days. But this will give you a working set of information on how far completed you are.

Now don't forget to pad the estimates — not because you are lazy or because you are trying to add in nap-time, but just because of the inevitable scope creep. There are going to be missteps and additional things added. Also the further you get in the development process, the more the programmers are spending time supporting the features (and bugs) that they've already created, and the less time they have at just pounding in new features. So remember that the rate of development starts fast, and continually slows down through the process. This gets worse once you've actually shipped — because now you have customer support and those communication issues to deal with — and of course the more bodies you add to the process, the more communication overhead there is (and things get slower as well).

The last thing to do is factor in all the final polish. Deployment is going to take time and resources. All the IS stuff, like setting up the machines, the servers, tweaking them, figuring out how to get everything talking, and so on, can be time consuming efforts. And then once you are ready to deploy there are other major efforts in profiling the App and

QUALITY

Experienced engineers and managers know that quality will be a function of all the legs of the triangle — but mostly time. The more time you have to mature the product, the better it will be. The fewer features you put in a product, the more time you have to mature the features that will make the product. And the fewer people you have on a project, the less communications problems (and overhead) and collisions you will have — but fewer people requires more time.

Time over scope is the most important thing for quality. That is why I advocate iterative development. It allows you to get a small subset of the functionality in first, and get much of the early design done quicker (and getting you started faster) — thus buying you time (experience) for the rest of the project.

The features implemented in the first pass, will have a few more passes to mature — and they are completed early. Everyone learns the processes early, when they are simpler — thus saving time (and not having to learn the hard way). People learn the product (or a key subset of it), as early as possible. Documentation gets a head start by being able to document the first iteration of the product (while the next iteration is being developed). QA (Quality Assurance), gets to be assuring the quality of the first iteration, while development is going on. Analysis and Design scales up in complexity only as it needs to — just-in-time for each pass, instead of having to do it all up front. You get to do test deployments of your solution earlier rather than later — thus you get over those learning curves as soon as possible.

This is all why three small steps are better than one big one — and why you don't want to over design and over implement in the first pass. You want to allow things to mature as they go — and give yourself more time (more stages) to go through every step of the process. By the time you release an iterative development product, it is really a 3.0 product, (assuming three iterations), instead of being a 1.0 version (with all the mistakes and issues being captured in one release, that you will have to fix in the next version)..

MT

figuring out where the bottlenecks are (or are going to be).

The total time estimate is just the count of all the time for all the functions. That should give you man days (or man months) of work — and you can use that to estimate how many people the project will take for a given time. But remember, time is quality. The more time the project has, the better the results. A few people on a project is good — it gives you a diversity of ideas — but beyond a few, the rate of development does not increase much. And there are latency versus throughput issues. Some things just have to be completed before others can be done — it is critical to look for these things. Nine women can't have a baby in one month. You want to get started on as many parts of the project as is possible as early as is possible.



It just takes one bug...

**The sooner you find that last bug,
the sooner you can ship,
and the sooner you can sleep.**

**Developer
DEPOT[®]**

Find these and other essential developer tools at

Spotlight 1.0



only \$189!

"Automatic Debugging" on the Macintosh. Easy to use, Spotlight can automatically find run time bugs in your code without your having to change one line of your code. Nail down random bugs immediately. Ever dereference the wrong pointer? Ever pass the wrong value to a toolbox command or over write an array? Your compiler often lets the code compile fine and you may not find that bug until you've burned it into CD-ROM or released it to the net. Find the bugs now, kill them easily, get to sleep sooner.

BugLink Solo



only \$79!

The only thing more frustrating than bugs, is bug reports. Users will send in everything from war and peace to "it crashed." BugLink Solo lets you define what information you want in a bug report, then include a customized BugReporter application with your application. Users simply click open the reporter, fill in the boxes, and click send. The report is transparently sent to the email address you defined. BugLink can login into that eMail account, download each report, and present you with an organized, complete, description of each bug. Great for shareware developers, system administrators, and web developers!

DCon 1.0

only \$39.95!

Every doctor knows, the secret is listening to the patient. The most frustrating part about debugging on Macintosh is you can't "printf" to show the state of a variable or position in your code. DCon lets your code talk to you. This system extension adds a console window and file logging services to Macintosh and can be used to record and display status and debugging information during development. It can be called from virtually any code, any where at any time - even from interrupt handlers, I/O completion routines, VBL tasks, deferred tasks, and more! DCon does not allocate memory in any Mac developers debugging arsenal.

www.devdepot.com

PO Box 5200 Westlake Village, CA 91359-5200 • Voice: 800/MACDEV-1 (800/622-3381)
Outside US/Canada: 805/494-9797 • Fax: 805/494-9798 • Email: orders@devdepot.com

Tip: Again, I tend to use the web to manage a web project. I put the time estimates on the web, often in an outline of the site itself — with total time and time towards completion (so far). Tracking how far you've come, and how far you have to go is pretty easy this way — and everyone can see progress and what is left.

IMPLEMENTATION

Implementation is where the rubber meets the road. It is where you have to do what you said you were going to do. This is much easier now that everyone has agreed on what it is you said you were going to do. You have the basics of a site map, where each page will be, and what is going to be on each page. You have the basics of what you have to store, and where that should be in the database. And you should have collected the basics of the common elements. You even have rough estimates on how much time each thing should take, so you can divide up the workload. Now get to work.

If you were really thinking ahead, you could have given yourself a head start. A trick here is that I use WebObjects to create most of the documentation and prototypes to get to this point, so that when it comes to implementation I have a working prototype and starter project.

When I create a site-map for each page and function — I do it using WebObjects itself. Thus the prototype is the documentation, and the prototype becomes the implementation. Just create a bunch of static HTML pages (using WebObjects), and use static links to take you from one page to the other. For the look, just include .gif or .jpeg snapshots of what each page will look like — and at the bottom of each HTML page, I just put the list of things that this page needs to do. When it becomes time to implement, I just snapshot this project, and I have a starter project with empty pages all ready to be filled in. As I implement each page I have a list of things (features) that need to be done, and I just remove from that list as I go. The documentation of what still has to be done can be right there (in the programmers face). You can even add in notes and issues to each page, for all those little things you don't want to forget (the developer notes on limitations of a function or page). And if you are fancy, with very little effort, you put these in a conditional so that you (developers or internal people) can show these notes are not — and you can keep the documentation and issues always in sync with the product.

The model work can be done the same way (on a smaller scale). I use EOModeler to collect a list of entities and attributes, and even show their relationships (sometimes). To create the documentation you can just generate the classes, and put in comments what each class or method needs to do (assuming you are putting business logic in the model, and not above it). Using Java Doc can be a big help here. When I'm done with the design phase, I can just tell EOModeler to generate the SQL to create the database.

As the design is maturing (as you go), you can even do other things. For global elements, create those elements (components) and what they need to do, and include them in or on the pages where they need to be. Then global elements are global in the documentation as well as in the design (and you just need to replace them with the functionality and they will work everywhere). For a list of support classes and middle layer business objects and methods, I just create place holders and shells using ProjectBuilder.

I was even thinking of getting fancy, and building a bug-tracking software right into the product itself (either the product or the shadow-product with the documentation) — so that QA could just turn on bugs using a back-door and add bugs right to the page that is having problems. And of course developers could view them and fix them right there as well. But there is a bit of work on that task (synchronization of pages and bugs, and good reporting abilities) — so I'm saving that one for a later effort.

There is still a whole lot of work to be done to finish the implementation— but at least these techniques can give you a head-start.

The goal is to make the documentation becomes the prototype which becomes the starting point for implementation. And later on the implementation can still contain parts of the documentation, and design notes, and other things. You can manage them in parallel, and keep the outline (with documentation, snapshots and project management, and bugs, etc.) separate, or put them in your actual project. There are strengths and weaknesses to both methods. On smaller projects I tend to keep them all in one, on larger ones, I've kept them in parallel — mostly because on larger projects I can afford to have someone else maintain the parallel effort, and it allows more diversification between versions. But the closer to the design you can keep the implementation (and vise versa), then the better you were at achieving your goals. Both ways result in getting a lot closer to your targets.

IN-OUT, REPEAT IF NECESSARY...

Remember, if you complete this process, you have only completed one iteration. That isn't finished, that is just the end of the beginning. So go back to the top, start over, and repeat "if necessary" — and it is always necessary.

WebObjects is a Rapid Application development tool. So don't fight the tool — use it. It is for prototyping (as well as deployment), so use it for the prototypes. Websites are good for dynamic documentation. WebObjects is good for building websites — so use it if it makes sense for you. At least consider using HTML and a website for your documentation — eat your own dog food. This all may seem blindingly obvious, but I'm surprised by how many companies (and individuals) don't capture their goals in simple design documents, or how many don't use prototyping or online documentation.

Analysis and design don't have to be hard. Most of the academic books written on these subjects have good ideas

in them, and good techniques — but can't be followed to the letter in the real world. Most of them are such overkill (for the commercial world) and so complex that people either waste too much time trying to do everything, they get frustrated by the books and all the things they have to know so they do nothing, or most often they try to do the former (and do everything) and burn out half way through and give up or say, "good enough" (and end up with nothing). Be pragmatic and use what works — for me, it has been starting at the top, and making a thin shell, and then filling it in and maturing it over time. In Aerospace, BioMed, or some financial institutions, there is so much risk to what programmers are doing, that they can afford huge time wasting processes and reduced development efficiency (and larger head-counts) for the sake of liability and documentation. Most commercial organizations can not afford that. Know your market, and what makes sense for your company and market.

Most companies try to separate development from project management, and apply abstract project management tools and layouts to a development task. It doesn't have to be that hard — just collect the requirements (scope), and layout how much time each thing will take, and add it up. You'll miss, but you'll usually be much closer than if you are basing your estimates on more abstract techniques. The closer you are to development (in the time estimation process) the closer you are likely to be to the target, and the better your estimations are likely to be.

It is amazing how many projects (and managers) still try to still do waterfall processes. They try to do all the design up front — as if we are omniscient beings — and they ignore how back-loaded the process is. You slow down as you go, because as you go you need to keep adding in more and more things — like documentation, support, bug fixing (and bug hunting), and so on. You also need to adapt to change as you go, and accept that feature creep is going to happen. Don't blame — try to capture the change request for a future version or reprioritize and trade-off functions — but accept change and that you can't think of everything before you start.

It takes many tries to get things right. Instead of trying to do it all right the first time, just accept the iterations and pick the low-hanging fruit. Layer the development (and analysis, and design) into iterations. This makes things simpler early on — and so you make quick progress (thus making everyone feel happy and productive). This also teaches everyone good habits (the process). The problems you think you're going to have may not materialize, the ones you are likely to have you weren't going to think of anyway. Don't design for features that may come down the pipe in years (or that you'll never have time to get to) — redesign when you need them, you'll know more at that time, and you'll have learned more during the time in between, so are likely to make better decisions later anyway. So by starting simple, you are training people on something easy, so that in later iterations, as things get

more complex, everyone is trained and comfortable on the process. When later (tougher) problems come up, everyone is more skilled with the process, as well as understanding the solution — so they are more ready to respond to any problems that arise.

Lastly, by doing iterations, you always have the safety net of previous iteration to fall back on (if you are late to delivery). You not only have a more accurate way of measuring how far you've come, how much is left to do, and how long each iteration takes — but you will have addressed the most important functions first, so they have the most time to mature. And while earlier versions may not be as full functional as the goals, they will be working versions which you can ship if need be — thus reducing risk to the company.

Doing things in layers (iterations) is faster, easier, makes everyone see progress and get more motivated. You are able to show others (investors or customers) the product sooner, and even sell it sooner. And it gives you more time to develop and mature the product — thus meaning a better product, for less time and less effort. Which is what the whole development process is about. Fortunately, WebObjects as a tool, can help make many steps of the process easier as well, if you use the tool for what it is good for.

MT

DATA RECOVERY: 800-440-1904

"Their incredibly kind staff focus on getting the job done as well as their customer's feelings. All experiences should be so pleasant."

—Neil Ticktin, Publisher
MacTech Magazine

7 Good Reasons to Choose DriveSavers



"We Can Save It!"

INTL: 415-382-2000
www.drivesavers.com

1. Fastest, most successful data recovery service available.
2. Retrieve recovered data instantly with DATAEXPRESS™ over high speed secured Internet lines.
3. Authorized to perform Data Recovery on all Apple computers and hard drives.
4. 24-hour, onsite, and weekend services.
5. Advanced, proprietary recovery techniques.
6. Featured by CNN, BBC, Forbes. Also in Mac World, Mac Addict, Popular Mechanics, and many others.
7. Federal and State Contracts (GSA, CMAS).



Since 1985

DriveSavers – Your Data Recovery Solution!

©2000 DRIVESAVERS, INC. 400 BEL MARIN KEYS BLVD., NOVATO, CA 94949, INTL: 415-382-2000, FAX: 415-883-0780

By Andrew C. Stone

An Intro to Mac OS X's Command Line Interface

Back in the mid to late eighties, DOS was king and PC users would make fun of the "MacInToy" simply because it had no expert command line interface (CLI). Of course, this was before the PC world ditched their own command line interface in favor of an uglified-down graphical user interface that poorly copied the Macintosh's finely detailed GUI. Being an old Unix weenie from the academic Computer Science scene, I can understand the argument that the command line is an extremely powerful tool for expert users and programmers. Of course I also understand that it needs to be hidden from the casual user, lest they type `rm -rf /` as root and totally erase their hard disk!

And now, the CLI is just one more compelling feature of Mac OS X. OS X is based on Darwin, Apple's version of the open source FreeBSD 3.x combined with the Carnegie Mellon University Mach 3.x kernel. The command line and all the standard Unix utilities are part and parcel of the developer distribution. While it is still not clear what will or won't ship to the end user in the final OS X, DP4 gives you Terminal.app (`/System/Administration/Terminal.app`) to provide a "shell" allowing you to type commands directly to the computer for both interactive and batch processing. This article is an

introduction into the use of the command line, and I hope to provide enough information to convince unrepentant GUI freaks that a CLI is indeed useful, and share some cool tricks, tips and information on where to go to learn more.

SHE SELLS C-SHELLS

Once upon a time, there was just the c-shell, `/bin/csh`. However, Unix hackers love to cram more and more features into any one given program, so now there are several shell programs to pick from, including: `tcsh`, `csh`, `zsh` and `bash`. When you double-click Terminal.app to launch a shell, Terminal uses your default shell as the command line interface. In a standard installation, the default shell is `tcsh`. Your default shell is stored in NetInfo, and can be set by editing the User in `/System/Administration/NetworkManager.app`. Click on "Users", then select a user, then edit the shell field to the complete path of your desired default shell. You can always switch shells on the fly by just typing the name of new shell. Each shell has its strengths and supporters - you might play around to see which one you prefer. I use `bash` whenever I need to be able to include newlines (returns) in a command - and it handles filenames with embedded spaces rather well. Each of the various shells share a lot in common, but syntax does vary from shell to shell, so I'll just use the most basic examples here which at least apply to `csh`. In any regard, launch Terminal.app so you can start to play!

Man - the manual pages

One of the Unix shell's most useful features is the builtin manual pages, which document all the commands. Before we learn about "man", a short discussion of why Unix uses such cryptic and abbreviated names is in order! If you have some familiarity with DOS, you'll know that to copy a file, you type:

```
copy <FILE_1> <FILE_2>
```

Andrew Stone <andrew@stone.com> is founder of Stone Design Corp <<http://www.stone.com/>> and has spent 12 years writing Cocoa software in its various incarnations.

The same command in Unix is "cp". It seems early Unix programmers hated to type and loved lacunae - and you'll learn Unix shell programming faster if you can uncover the mnemonic embedded in each command. For example, csh = C shell; mv = move; ln = link; ls = list; mkdir = make directory; etc. But the cool thing is that you don't have to remember much to use Unix because the man pages detail the use of every Unix command. For example, to learn about the manual pages, type:

```
man man
```

You'll see something like:

```
MAN(1)  System Reference Manual  MAN(1)
```

NAME

```
man - display the on-line manual pages
```

SYNOPSIS

```
man [-achw] [-C file] [-M path] [-m path] [section] name ...
```

DESCRIPTION

```
The man utility displays the BSD Unix manual pages
entitled name.
```

The options are as follows:

```
-a Display all of the manual pages for a
specified section and name combination.
(Normally, only the first manual page found is displayed.)
....
```

When you can't remember the command that you want to learn about, you can always try the apropos command, which searches the man pages' names for the given word. For example, typing

```
apropos directory
```

gives back a list of Unix commands whose description includes "directory" (directory is the Unix term for folder).

```
DirHandle(3)      - supply object methods for directory handles
FindBin(3)        - Locate directory of original perl script
Tcl_TranslateFileName(3) - convert file name to native form
                        and replace tilde with home
                        directory
basename(1), dirname(1) - return filename or directory
                        portion of pathname
cd(1)             - change working directory
cd(n)             - Change working directory
chdir(2), fchdir(2) - change current working directory
chroot(2)         - change root directory
chroot(8)         - change root directory
dir(5), dirent(5) - directory file format
directory(3), opendir(3), readdir(3), telldir(3), seekdir(3),
rewinddir(3), closedir(3), dirfd(3) - directory operations
ditto(8)          - copy source directories to destination
                        directory
getcwd(3)         - get pathname of current working directory
getcwd(3), getwd(3) - get working directory pathname
getdirent(2)      - get directory entries in a filesystem
                        independent format
ls(1)             - list directory contents
mkdir(2)          - make a directory file
mtree(8)          - map a directory hierarchy
nfind(1)          - find a directory in the NetInfo hierarchy
pax(1)            - read and write file archives and copy
                        directory hierarchies
```

Mac OS X

Porting & Development Showcase

Mac OS X

Making the Move

With **MAC OS X** just over the **HORIZON**,

Mac **developers** everywhere have a major need for **CARBON** and **COCOA** application porting, **AQUA** user interface implementation, and **DRIVER** development services. Toward that end, several high-quality **SOFTWARE** engineering firms, in association with the **APPLE DEVELOPER CONNECTION**, are offering these services at very **attractive** discounts to all **ADC** Select and Premier members.

The following pages
are those vendors that
are part of the Mac OS X

Porting & Development Showcase.

Programming at the Speed of Light

Aqua

Cross-Platform

Device Drivers

TCP / IP

Graphics

Plug-Ins

Mac/Unix/Win32

Ports

Carbon / OS X



Software Development
Outsourcing Since 1990

415.491.2200
www.shadetreeinc.com

Get the power and experience you need from

PROSOFT

e n g i n e e r i n g , i n c .

With over thirteen years in providing programming service to the Mac community. Prosoft is committed in delivering the ultimate quality software products and service.

Our Engineers have extensive knowledge in:

- Macintosh PPC Drivers, Shims and Extensions
- Macintosh Power Plant Applications, Mac OS X applications and drivers
- Carbon application porting for Mac OS X
- Multimedia and QuickTime Applications
- Unix/Linux Console Applications, including Unix Solaris, xBSD Drivers, and Linux Drivers

Other areas of expertise

- Windows 95, 98
- Window CE
- Windows NT, 2000
- Unix / Linux
- Java
- Embedded OS
- RDBMS
- Network Consulting

- ☐ Eat your vegetables.
- ☐ Exercise every day.
- ☒ Port to Mac OS X.
- ☐ Call your mom.

All of these are good for you.
We can make one of them easy.

Since 1989, The Omni Group has worked with the technologies that have been refined into Mac OS X.

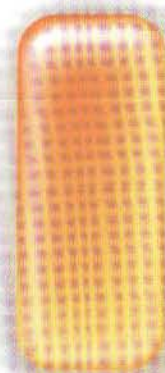
Moving to Mac OS X is a big step for your company, and you need consultants who can help you both plan how best to make the transition and follow whatever path you choose.

That's been our business for years. No matter what kind of product you have, we can get it up under OS X, fast:

- Real games: We ported id's Doom and Quake games to NEXTSTEP and OS X. Quake 2 took us a week.
- Big apps: We ported Adobe's FrameMaker to NEXTSTEP and Sun's Concurrency to OpenStep/Solaris.
- Big libraries: We ported the Oracle 8 client libraries which Apple ships today in OS X Server.
- Serious drivers: We ported 3dfx's Glide and wrote Voodoo2 and Rendition drivers for OS X Server.
- New apps: We wrote OmniWeb, the only native OS X web browser, and OmniPDF, the native Acrobat viewer for OS X.

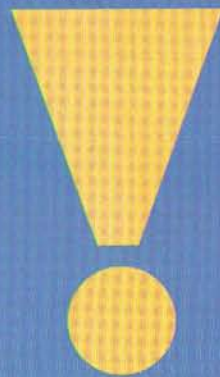
Mac OS X is what we do. Let us help you do it, too.

THE OMNI GROUP



2707 Northeast Blakeley Street
Seattle, Washington 98105-3118
www.omnigroup.com/consulting

sales@omnigroup.com
800.315.OMNI x201
206.523.4152 x201



makers of

PERIPHERALS, EXPANSION CARDS & SMART DEVICES

get OS X compatible

Since 1991, high tech companies like Apple Computer, Hewlett Packard, and Leica Microscopy have turned to Art & Logic for assistance in implementing cutting edge technologies.

Now, with the advent of OS X, Art & Logic is helping companies make their products compatible with Apple's new operating system. Art & Logic engineers are industry leaders in Carbon & Cocoa porting, Aqua implementation, and driver development.

In the new economy, where time to market is everything, you need results. Art & Logic delivers.

"We have found the engineers at Art & Logic to be outstanding—better than excellent. When we use their software development services, we get results."

Ted Dykes, CEO of LRO, Inc.



Art & Logic, Inc.

www.artlogic.com

877-278-5644 (toll free)

info@artlogic.com

The software engineering company that helps bring your hardware product to market—guaranteed.



get to

Mac OS X

fast!

*With the experienced
Mac OS development team.*

Red Rock Software specializes in the Macintosh software development. Red Rock Software's team of senior engineers averages over 10 years of development experience on the Macintosh platform. Our expertise and experience has gained us the trust of companies like *Apple Computer, Iomega Corporation and Sorenson Media.*

If you are looking to get your product compatible with Mac OS X quickly and with extreme quality, let our experts help.

- Aqua Interface Implementation
- OS X Carbon Porting
- OS X Native Cocoa Application Development

R E D R O C K

s o f t w a r e

call Red Rock at 888.689.3038
or visit us online at www.redrocksw.com



Robosoft Technologies

www.robosoftin.com

partners
in product
development

*Indian mind power@
an unbeatable price*

Carbon Porting
Aqua Implementation
Windows to Mac Porting

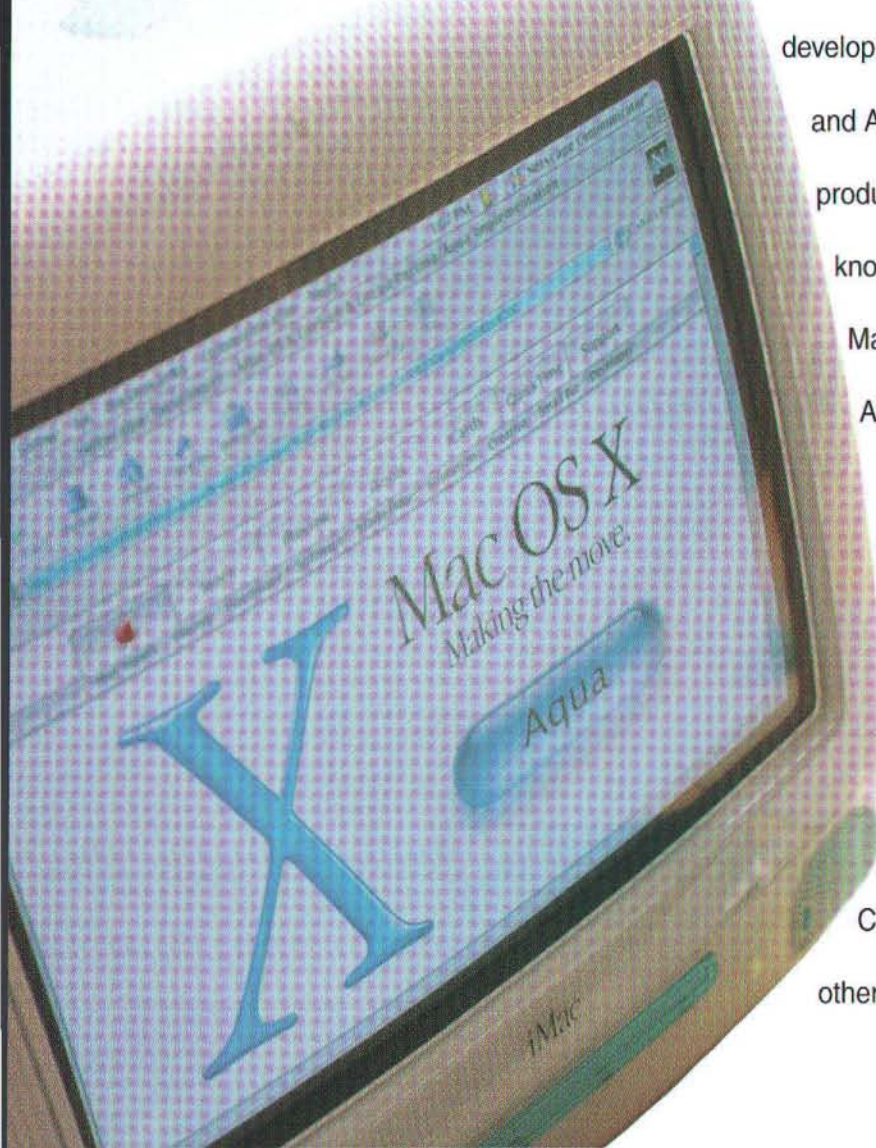
Special discounts for ADC members

Robosoft is an India based specialist in Mac product development, Windows to Mac porting, Carbon porting and Aqua implementation. Our engineers have a strong product development experience with a thorough knowledge of C/C++, Java, Mac OS Toolbox, Macsbug, QuickTime, Carbon API and PowerPlant Application Framework.

We have a proven track record for product development out sourcing and delivering the projects on time. Our clients include a veritable who's who in the industry - Apple, Adobe, Quark, FileWave, Versaware, Veon, NetJumper, CEI, The Anderson Group and eCapital, among others. To know us better, visit our website

>>>>>

info@robosoftin.com




```

pwd(1)      - return working directory name
pwd(n)      - Return the current working directory
remove(3)   - remove directory entry
rm(1)       - remove directory entries
rmdir(2)    - remove a directory file
scandir(3), alphasort(3) - scan a directory
unlink(2)   - remove directory entry
whois(1)    - Internet user name directory service

```

Apropos reveals the rich set of commands available to both the command line user and C programmer. The number following the command indicates which man subdirectory should be consulted. To read about a man page in a particular subsection, for example, scandir(3), you'd type:

```
man 3 scandir
```

If you just typed that in, you'll notice that it's actually a C language prototype, and not really available as a command line call! If the subdirectory is 1, it's a shell command; otherwise it's a C function. Still, most standard C library functions are exposed as callable binary programs or shell scripts.

ANATOMY OF A SHELL COMMAND

After looking at a few Unix man pages, you'll notice that a consistent style is used to make it easy to grasp the functionality and options of a **command**. The simplest example is a command which takes no flags or arguments, for example:

```
date
```

A **flag** modifies how the command works and is represented by a letter or digit which follows a "-" (dash) or another flag. **Arguments** are input to the command, such as filenames or folders:

```
cp -rf /bin/cat /tmp/catty
command flags argument 1 argument 2
```

The man page presents a synopsis of the command in a standardized notation to express when a flag or argument is optional and/or repeatable. Let's look at the first example of the synopsis of "man" to learn how to decode the notation:

```
man [-achw] [-C file] [-M path] [-m path] [section] name ...
```

Anything enclosed within "[" and "]" represents optional flags and arguments. For example you could include -a or -c or -ac or -ca or -achw as valid flags to the man command.

When a word follows a flag, you must provide an argument if you use that flag. For example [-M path] means that if you use the optional -M flag, you must follow it with a full path name.

If you see ellipses, such as at the end of the synopsis, it means you can provide any number of additional names as arguments to the command.

All of the flags and arguments are described in the body of the man page for the command. Also, at the end of the man page is a section which gives additional related commands. For instance, on man's man page, you would find:

```
SEE ALSO
apropos(1), whatis(1), whereis(1), man.conf(5), mdoc(7),
mdoc.samples(7)
```

Do a "man" on some of these to keep learning more.

LESS TYPING, MORE POWER

Power users, who are always looking for ways to maximize the amount accomplished with a minimal amount of typing, use **initialization files** to personalize their shells. Initialization files contain environmental variables, aliases, and other customizations, such as a clever shell prompt. Each shell program looks for a different initialization file; for example, csh uses ~/.cshrc and bash uses ~/.bashrc. *Extra Credit:* using man, find the name of the initialization file for tsch!

Aliases are short cuts to make complex commands that you use available with just a few typed characters. For example, let's say you always want to see your directory files listed in reverse order of the time created, that is, with the newest files listed at the top with the long listing:

```
ls -lt
```

You could add this line to your ~/.cshrc file:

```
alias lt "ls -lt"
```

Next time you start a c shell (or type the command "source ~/.cshrc" to reread the initialization file), when you type "lt", you'll get the equivalent of typing "ls -lt". People love to put gnarly prompt creation commands in their init files, and this example is about as scary as they get:

```
alias cd 'cd \!*; set currDir = `basename $cwd`; set currDir =
`echo "<${host}:$currDir
" ! >"; set prompt = "$currDir "'
cd $cwd
```

This produces a lovely, palatable prompt which tells you which machine you are on, which directory you are in, and the number of the current command, for example:

```
<hermione:Developer 24 >
```


Which leads us to a discussion of typing less through use of history and command substitution. Your "history" is a record of the commands you've issued. Type

```
history
```

to see a numbered list of the commands you have typed recently:

```
1 cd
2 cd Library/Preferences/
3 rm ApplicationCacheTheFinalChapter
4 ps -axw | grep Mail
5 man man
6 apropos link
7 apropos time
8 apropos link
9 apropos directory
```

To repeat a previous command, for example, "man man", simply type:

```
!!
```

!! is pronounced "BANG-5" in Unix. A "*" is "splat". A "|" is a pipe, although French Surrealist painter Ren Magritte would say, "Ce n'est pas un pipe"!

To repeat the last command issued, type:

```
!!
```

To refer to the last argument of the last command, use"!\$". For example:

```
cat !$
```

displays the contents of the last file you referenced in your last command.

In some shells, you can use your up and down arrow keys to repeat commands issued before.

Because many shell commands operate on files, and because typing file names is tedious and prone to error, you have various shortcut options.

Most shells have builtin "escape completion" which will complete the names of files and folders if you type the first few unique characters and then type your escape completion key (this might be "ESC" or F5 or even TAB in some shells). The vanilla C shell has this turned off by default, but you can add the following line to your .cshrc file to enable escape completion:

```
set filec
```

As of DP4, the default shell is tsch, which has escape completion on. Type two escapes if one escape doesn't complete anything.

And, for you GUI junkies, OS X lets you drag and drop files or folders from the Finder onto shells, instead of laboriously typing out entire path names. You'll even get the correct escape codes for spaces, or

www.macshowlive.com

EVERYTHING* MAC

EVERY WEEK

LIVE!

Each Wednesday Night

join us for:

- Mac News Review
- Featured Live Guests
- Regular segments on:
 - Gaming
 - Basics
 - Tech Tips

And audience

interactivity!

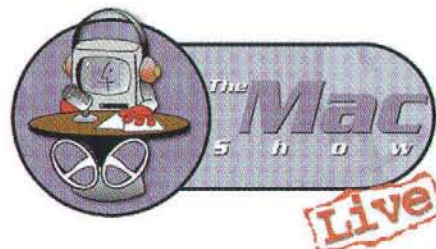
- Java and IRC Chat
- Toll-Free Phone-in
- Contests and prizes

To listen, you'll need:

QuickTime 4, a Modem and a

Mac

**almost*



Every Wednesday 9-11PM EST or listen to a stream or download archive of the show anytime!

www.macshowlive.com

perhaps the whole path will be correctly quoted. Because the shell interprets "words" (alphanumeric strings separated by spaces or tabs), you need to use to either "escape" the characters with a backslash:

```
cd /Mac\ OS\ X
```

or type it with quotes:

```
cd "/Mac OS X"
```

BULK PROCESSING AND ITERATIONS

Where a CLI has it all over a GUI is when you need to perform some boring, repetitive task on a bunch of files. For example, suppose you want to change the path extension of a whole folder full of files, say from ".TIF" to ".tiff". You can't do this in a GUI without an additional program, but in the C shell, you can use a simple looping shell command, using the "foreach" statement. Type:

```
foreach i ( *.TIF )
```

At this point, you enter an interactive subshell with a new prompt, "? ". You can type any number of commands for sequential processing, referring to the current file as \$i, and the "root" of that file (\$i:r), ie, the filename's full path removing its path extension. See "man cshrc" for more info on referring to parts of a file path. Finally, when you are finished issuing commands, type end and computation begins.

```
? echo $i           // which file are we on? We'll print it out
? mv $i $i:r.tiff   // move that file to the same name, less extension, plus .tiff
? end               // this terminates the loop and begins processing
```

PLUMBING: INPUT, OUTPUT AND PIPES

Unix is designed for hooking up the output from one process to the input of another. This is done through two basic mechanisms: redirection of standard input/standard output and pipes.

The less than symbol, "<", means take input from whatever is producing output on the right hand side of the "<". For example, you can use the command line "mail" to send ascii files:

```
mail harry@stone.com < /etc/hostconfig
```

You can use I/O redirection whenever a command expects standard input (stdin) or produces standard output (stdout). When a process expects a feed of input, then you must rely on the chaining of processes through pipes. Pipes can be used effectively with gnutar to copy large trees of files to new locations:

```
(/usr/bin/gnutar chf - Stone_CD) | (cd /Backup;
/usr/bin/gnutar xf -)
```

Note how you can also stage commands by separating them with a ";". Here, we are creating a new "tar" archive, changing to a new directory, and then unarchiving it to standard out, thus copying the original hierarchy of files to this new location.

Once you get the hang of Unix, you'll find that you can do all sorts of tasks. For example, the age old bean counting task of counting lines of code can be done trivially using the powerful and confusing "find" piped to the meta-command "xargs" which calls the word count program "wc". Assuming you have changed directory to the top level of your source code, type:

```
find . -name '*.hmc' | xargs wc -l
```

which translates to: beginning at the top level in this current directory, recursively find all files which end in either .h, .c or .m, and use each file found as the input to the word counting program, with the option of counting only lines (not words or characters), with the bonus of providing a grand total of lines. What may look like jibberish will soon become recognizable and habitual!

And, for authors paid by the word and to honor the oft times obfuscated nature of Unix, here's an alias for your initialization file:

```
alias wordcount
(cat !* | tr -s '...:?!()[]"' '\012' | cat -n | tail -
1 | awk '{print $1}')
```


and you would use it in a source directory by typing:

```
wordcount *.hmc
```

To go "deeper" you just add on more arguments:

```
wordcount *.hmc */*[hmc] */*[hmc]
```

CONCLUSION

It takes years to learn Unix completely, but only moments to learn enough to be useful. The online man pages allow a quick refresher course in any command and are an excellent lifetime teaching tool. The shell is a power user's friend and can expand the capabilities and productivity of OS X. Granted that it is not for everyone, but for those who dare, it's a true bonus! 

Want to share a tip with the
community and get paid for it?
Send it in to
tips@mactech.com

ANY FASTER AND
THEY'D HAVE THE
SOFTWARE BEFORE
THEY BUY IT.



e sellerate
The new way to sell software

MindVision Software, creator of Installer VISE, introduces eSellerate, the quick and easy way to speed up your online sales. Designed especially for developers like you, eSellerate puts instant gratification into the hands of your customers. Now, your application can sell itself with no manual intervention from you. None, nothing. Nada. www.esellerate.net

By Tom Djajadiningrat

Satimage's Smile

AppleScript IDE

INTRODUCTION

Smile is an AppleScript integrated development environment which is highly scriptable itself. It offers a fine text editor, pretty good documentation and the possibility to look up definitions from within the editor. But Smile's most amazing features are its ability to execute a script line-by-line from a seemingly ordinary text window and that its interface can be customized at will.

Since it is essentially free — all SatImage asks is your feedback to further improve Smile — it makes a great replacement for the standard Apple Script Editor. After a while you realize that Smile is much more than that. It brings together in a single application a combination of features which force you to rethink the possibilities of AppleScript.

REQUIREMENTS

- Smile (downloadable from <http://www.tandb.com.au/smile/> and <http://www.users.imaginet.fr/~satimage/>.)
- A PowerMacintosh (Smile does not run on 68K machines.)
- MacOS 8.5 (unless you are willing to mess about with the Navigation Services and Appearance extensions on earlier systems.)

A SUPERIOR EDITOR

Despite its name, the editing features of Apple's Script Editor have always been somewhat poor. Especially its lack of drag-and drop and search-and-replace is rather annoying. Smile is worth getting just for these features. In addition, Smile features live scrolling, can compare files, has no 32k size limit, and implements Appearance. Command clicking the title bar of a window gives CodeWarrior-like access to its file's path (**Figure 1**). This in combination with support for Navigation Services makes organizing files much easier.

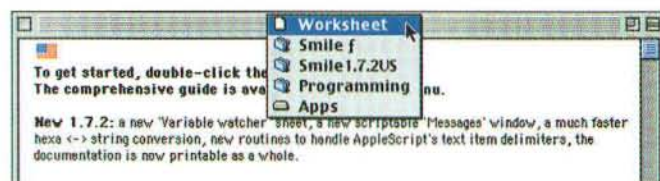


Figure 1. Smile provides easy access to a file's path.

If you are a keyboard animal then you will appreciate that keyboard navigation and selection are fully implemented. Jump word (option + left/right arrow key), jump start/end of line (command + left/right arrow key), page up/down (option + up/down arrow key) and home/end of document (command + up/down arrow key) and the selection equivalents with the shift key held down: it's all there.

DOCUMENTATION

Smile comes with a guided tour in its own file format. It is clear and concise, taking about 10-15 minutes to work through. More elaborate documentation can be found within the Help menu. Smile's documentation makes use of Apple's Help Viewer and turns up as an option within the Help Center (**Figure 2**). If you get totally stuck with Smile you can subscribe to the Smile mailing list as described in the help.

Since being involved in the re-design of the id-StudioLab web pages <www.io.tudelft.nl/id-studiolab/djajadiningrat/> **Tom** has got an even stronger opinion about layout and HTML. If you enjoy getting empty mail messages ask him for it with all four letter words left out at <J.P.Djajadiningrat@io.tudelft.nl>.



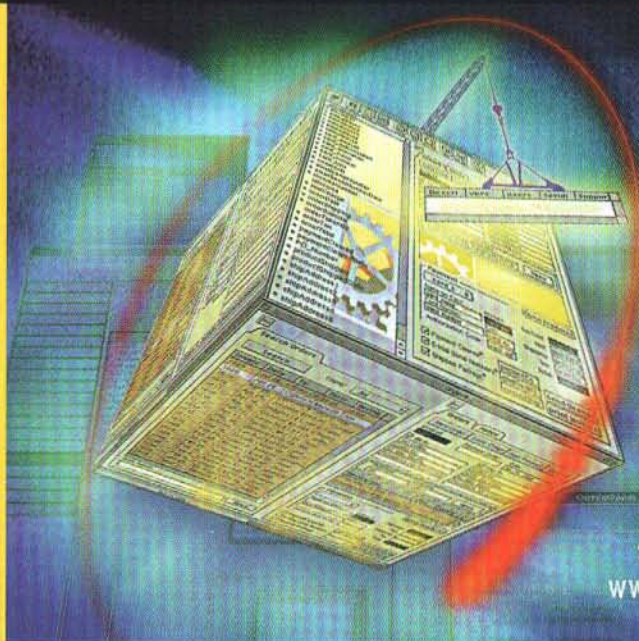
OPEN BASE RADSTUDIO™

Java
Enabled!

RAPID APPLICATION DEVELOPMENT ENVIRONMENT

Building
Database
Applications
Has
Never Been
So Fast!

Powered By
OPENBASE SQL



OpenBase RADstudio — a rapid application development environment for building Java-enabled database applications.

Easy Drag & Drop Tools. Intuitive drag and drop GUI building tools and event driven OpenScript 4GL accelerate application development for software designers and end users alike.

Instant Deployment. Applications developed with RADstudio are stored in a central database. Users always access the latest software versions.

Scalable Performance. RADstudio is powered by OpenBase SQL, offering high-performance data retrieval and transaction control for demanding multi-user environments.

See RADstudio today!

www.openbase.com/RAD



Figure 2. Smile's help turns up in the Apple Help Center.

A DIFFERENT PHILOSOPHY

If you like you can write scripts straight within a script window just as in the Script Editor. However, there is a far more powerful way to use Smile. Scripts can be written in a text window and a single line or piece of the code can be executed simply by selecting it and pressing enter (For those familiar with Mathematica this is somewhat similar to selecting cells and pressing enter after each one). This allows much flexibility during development as you can execute pieces of script in an

order of your choosing. It also allows you to leave pieces of code hanging about without the need to comment them out. Once a script works to your satisfaction, you can copy it to a script window and save it in the usual variety of formats.

Apart from script windows and text windows Smile uses a worksheet and output windows, making four different types of window in total. **Figure 3** shows all four at the same time. Let's have at each of them.

Worksheet

This is like a single page Scrapbook particular to Smile. Here you can store text and images which you use over and over again in different scripts, such as a copyright notice, those code samples you can never remember or logos. The result of executing a script in a script window is appended to the worksheet. The worksheet is saved automatically on quit and opened automatically when you start up Smile.

Text Window

During script development a text window is like a scratch pad. In a text window you can try out pieces of code in any order. You can have multiple text windows holding different versions of your code. When a piece of script in a text window is executed, the result shows up at the bottom of the same window. That is, unless you specify an...

Output Window

Selecting a text window and choosing Output window from the script menu creates a new window and redirects the output to that new window. You can also link an output window to a script window instead of to a text window. Then the output of the script window does not end up in the worksheet but in the output window.

Script Window

As mentioned before, you move your script to a script window once you wish to save it as an applet. The script window distinguishes itself visually by a comments area at the top, a coloured background, and a small toolbar at the bottom. Unlike a text window a script window does not allow execution of code line by line, it only allows you to run a whole script.

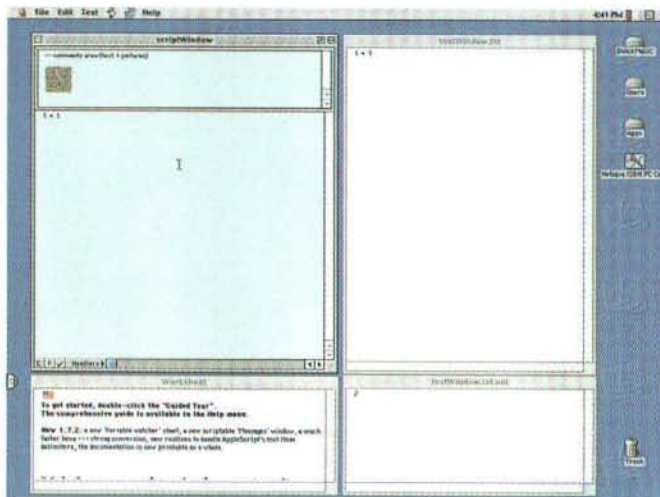


Figure 3. The four types of window in Smile: script window (top left), text window (top right), output window (bottom right) and worksheet (bottom left).

DECENT DEBUGGING

'Step-by-step' debugging

Let's go back to executing code in a text window. Putting the cursor on a line and pressing enter executes that line of code and advances the cursor to the start of the next line. So by pressing enter repeatedly you can step through the code. It even works properly with lines that are broken with option-return (⌘-return). Note how we almost have step-by-step debugging here. Well, almost but not quite. You need to watch out for a few complications.

If you use a tell block to direct commands to a particular application you can't just step into the tell block: Smile will choke on the first line tell application "x" and display an error message (Figure 4). It does work if you select the whole tell block and press enter, but of course you lose resolution: if things go wrong you don't know which line within the block

formed the culprit. Another option is to add tell application "x" to every line though this is a somewhat cumbersome solution.



Figure 4. Stepping into a tell block brings up an error message

This is where the tell command from the edit menu comes in (Figure 5). It lets you link a Smile text window to a particular application so that commands are directed at that application. Choosing the tell command makes a dialog box appear in which you can choose an application that is running on either the local or a remote machine (Figure 6). This turns on a small popup menu in the bottom left corner of a text window (Figure 7). Any script commands are now directed to the application you just linked to unless you specifically ask them to be directed elsewhere. So you no longer need to wrap the commands for the linked app in a tell block or use a tell command on every line. Choosing logout from the popup menu breaks the link to the application (Figure 7).

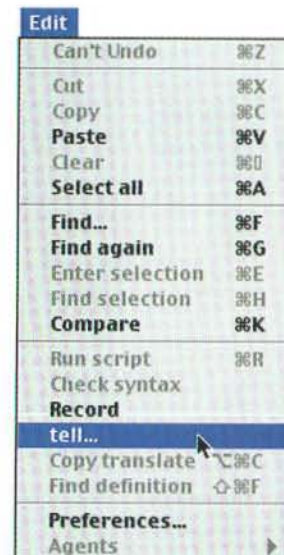


Figure 5. Use the tell command to direct AppleScript commands to a particular application.



"I registered my sharp idea"

www.engardefitness.com

Sharon Monplaisir, Olympic fencer and entrepreneur, registered her domain name to put her business online at register.com. Visit us at www.register.com or call us at 1-800-7-WWW-NET, and we'll help you register your domain name right over the phone.

register
● **com**TM
the *first* step on the webSM

Macworld

Conference & Expo

ahead

To Request
More Information,
Visit www.macworldexpo.com!

Moscone Convention Center
San Francisco, CA
January 9 – 12, 2001

Register Online
www.macworldexpo.com

Call Toll Free
1-800-645-EXPO

Register by December 11, 2000 for special savings!
Refer to Priority Code: **A-MTCH**

World-Class Exposition!
Macworld Conference & Expo/San Francisco 2001 is a One-Stop-Shop
offering discounts not found
anywhere else!

- Featuring more than 500 exhibiting companies
- Thousands of new products, software and services
- Hands-on demonstrations
- Test-drive new technologies
- Awards and competitions
- Prizes, drawings, giveaways

Special Interest Boulevards

Macworld Conference & Expo offers so much for every Mac user!

Check out the Special Interest Boulevards while you are exploring the exhibit floor to find and compare hot new products for your particular needs. Popular areas include:

- Digital Multimedia
- Music and Audio
- Developer Central
- Net Innovators

Be sure to visit www.macworldexpo.com for exciting additions!

**Join your friends
and colleagues in the Mac community
at The Largest IT event
on the West Coast!**

Cutting-Edge Educational Programs!

**Conference sessions for the
New, Beginning, Intermediate and
Advanced users!**

Macworld/Pro features technically in-depth presentations and issues-oriented discussions about professional applications of Macintosh technology. The Pro conference offers **the most sophisticated training available** on Mac networking, digital video, art director/creative management practices, and Mac systems administrations for large organizations.

Macworld/Users continues to be one of **the best educational values anywhere.**

The Users conference features industry experts offering skill development on the most popular Mac-related tools and professional development courses for users who rely on Apple technology to gain a competitive advantage.

MacBeginnings - San Francisco debut!

After great success this July in New York, MacBeginnings will make its debut in San Francisco! These high-energy, informative sessions will provide first-time attendees, new and beginning Mac users a starting point to enter the Mac community. MacBeginnings are free and open to **ALL** registered Macworld Conference & Expo attendees!

Macworld
Conference & Expo
www.macworldexpo.com

Flagship Sponsors

Macworld

MACBUY.COM

Macworld.com

MacWEEK.com

MacCentral.com

Owned and Managed by
IDG
WORLD EXPO

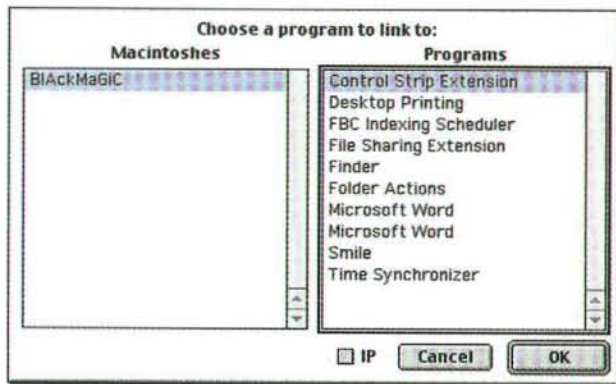


Figure 6. Specify the application to link to. Here only the local host with its processes is shown.

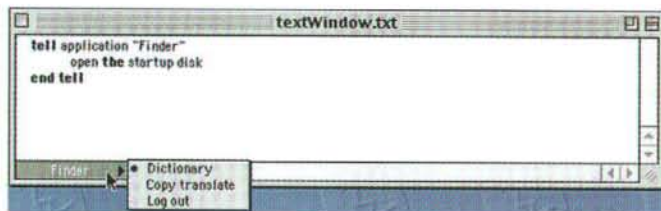


Figure 7. The tell-related popup menu.

Variable watching

To further facilitate debugging Smile offers a variable watching window. After dragging or typing in a variable name in this window Smile will show its type and value during execution (Figure 8).

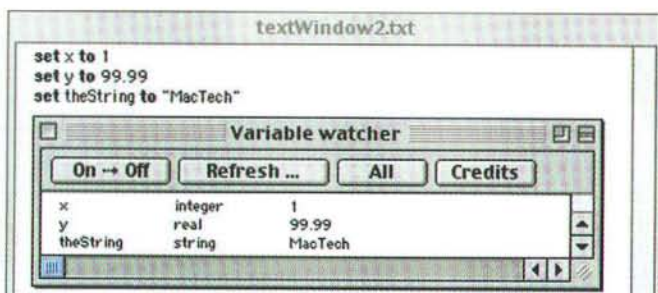


Figure 8: The variable watcher showing an integer, a real and a string.

HELP WITH APPLESCRIPT SYNTAX

In addition to the open dictionary command familiar from Script Editor, Smile has got some extra features to help you with AppleScript syntax problems. First, you can select Find Definition from the Edit menu which will try to find the string in Smile's dictionary (or if you just linked to another app, that app's dictionary) or in a scripting addition. Second, if you need help with the AppleScript syntax of an application that you are linked

to through the tell command you can choose Dictionary from the popup menu in the bottom left hand corner (Figure 7). I noticed that Smile chokes on the dictionary of Palm Desktop, while Script Editor has no problems with it. Finally, Smile's own dictionary is available under the Apple menu.

ADVANCED STUFF

Portability

Let's have a brief look at the somewhat more advanced stuff Smile has to offer. One nice feature is that it can assist you in writing portable code in which the normal English like AppleScript commands are replaced by raw codes. For example, if you want to script a particular application you may run into the problem that a particular user has a localized version of that app with a localized name (like SimpleText being called SimpelTekst in Dutch) or a version with a slightly different name (like BBEdit 5.x instead of 4.x). If you want the script to run smoothly and prevent a choose application dialog box from popping up you can look up the application by creator code instead of by name and resolve the name of the application at runtime. However, this means you cannot use the normal English-like commands as found in the app's dictionary, you will have to use the raw codes. If you first link to a particular application through the tell command you can then use the Copy Translate feature from the bottom left popup menu to change the selected text to raw codes (Figure 7).

Attachable and factored

Smile is not only scriptable and recordable, it also has those more rare AppleScript virtues: it is attachable, allowing scripts to be attached to its objects, and factored. What factored comes down to is that when the user chooses a certain command Smile sends an Apple event to itself instructing itself to execute the command rather than execute it directly. This ensures that Smile reacts in exactly the same way regardless of whether the Apple event originated from within or from outside Smile. This attachable and factored architecture is particularly powerful in combination with...

Custom interfaces

You can build your own custom interface for controlling Smile and AppleScripts. For example, you can add buttons to text windows to create so called Smile sheets and put up your own dialog boxes. Option-command clicking a button or dialog opens the attached script. Pretty impressive stuff. Even the Variable Watcher is really a Smile sheet. When it comes to customization you are pretty much on your own though as the documentation concerning this feature is rather skimpy.

INTERFACE CRITIQUE

Smile's interface could do with a little tidying up. Especially the script menu deserves some attention. At the moment it is a bit of a hodge-podge of all kinds of little tidbits provided by SatImage. As this is also the menu in which the user can place his own scripts, the structure of this menu can only get worse.

Then there is the linking of output windows to text or script windows. To me this seems a pretty essential component of the Smile philosophy, yet the command lives under the scripts menu. Since choosing output window essentially creates a special kind of text window the command would seem to be more at home under the file menu. I also found it a shame that the format of an output window is stubby and wide below the text window to which it is connected. The consequence is that any results quickly scroll out of view. I would prefer to have a tall and narrow output window, positioned to the side of the text window, so that older results remain visible and the most is made of the available screen real estate.

The Balance command's name does not do it justice. It suggests that the command is limited to indicating or closing unbalanced parentheses. However, it is in fact far more powerful than that. It can also suggest syntax, colourize indentation levels and assist you in writing tell statements, depending on what part of your script is selected before you activate it.

Finally, I find it a bit of a shame that a selection in a read-only document is not highlighted. The idea of having scripts embedded in read-only documents is quite tempting. It does not feel quite right though if you cannot see what you have selected before you press enter. Other applications, such as Tex-Edit Plus, do highlight selections within read-only documents.

MORE INFORMATION

While Smile seems to have gained quite a bit of momentum amongst hardcore AppleScript fans, it does not appear to be

very well known among the Mac community at large. Here are some URLs to help you explore further:

- <http://www.macuser.co.uk/burbulis/printreview.php3?id=35124> (one of the few mainstream reviews)
- <http://www.AppleScriptSourcebook.com/products/smile/smile.html> (in-depth review of Smile itself)
- <http://www.tandb.com.au/smile/pgs/SmileandScriptEditor.html> (Smile versus the Script Editor)
- <http://www.tandb.com.au/applescript/editors/> (thorough review of available AppleScript editors)

CONCLUSION

They say you can't look a gift horse in the mouth but Smile easily stands up to scrutiny. Most of my gripes concern the interface which deserves some more attention. All in all this is a great addition to the AppleScript scene though. The editing features, powerful lookup functions, line-by-line execution and variable watching make Smile far superior to Apple's Script Editor. The idea of system wide control from scripts which happily live together with styled text and graphics in a single document is fascinating. More advanced scripters will appreciate the support for portable code, the possibilities to script Smile itself and the customizability of its interface through Smile sheets and dialogs. The nicest thing about Smile is probably its gradual but seemingly insatiable learning curve: it is easy to pick up but it is difficult to imagine exhausting its possibilities. **BT**

Are you lost out there ?

Can't find anyone to help you get your product to market ?

WWW.BZZZZZZ.COM

BeHive Technologies Inc., The High Tech Cottage Industry Cooperative on the Internet

by Bob Boonstra, Westford, MA

WHAT BILLS DID THEY PAY?

This month we are helping the accounting department of a small business with a problem. Our little business is a daily newspaper that survives on its advertising revenue. Each day they print a newspaper, and each newspaper contains a number of ads. Whenever the accounting staff can find the time, they send out invoices to their advertising customers. Sometimes they bill daily, sometimes weekly, sometimes irregularly during a week or longer period. Most of the customers pay promptly, and most of them reference the invoice being paid along with their Payment. That's good, because our little newspaper has only a small accounting staff with no time to reconcile Payments.

The problem is with a couple of the newspaper's larger customers. These customers don't always pay promptly, and some of them don't reference the invoice when they pay. To make things worse, they sometimes pay part of an invoice, or pay multiple invoices with a simple payment.

Your job is to help sort this all out. The prototype for the code you should write is:

```
#include "OSUtils.h"

typedef struct Invoice {
    DateTimeRec invoiceDate; /* date on which customer was invoiced */
    long    invoiceAmount; /* amount customer was invoiced */
    long    invoiceNumber; /* reference invoice number */
} Invoice;

typedef struct Payment {
    DateTimeRec paymentDate; /* date payment was submitted */
    long    paymentAmount; /* amount of payment */
    long    paymentNumber; /* reference payment number */
} Payment;

typedef struct Reconciliation {
    long    paymentNumber; /* reference number of payment being applied */
    long    invoiceNumber; /* reference number of invoice to which payment is applied, zero if duplicate */
    long    appliedAmount; /* amount of referenced payment being applied to this invoice */
} Reconciliation;

long /* number of reconciliation records returned */ ReconcilePayments (
    const Invoice theInvoices[], /* invoices to reconcile, sorted by increasing date */
    const Payment thePayments[], /* payments to reconcile, sorted by increasing date */
    Reconciliation theReconciliation[], /* return reconciliation here */
    long numberOfReconciliationRecords, /* number of theReconciliation records preallocated */
    long *lateDollarDays /* see problem description */
);
```

The objective of this Challenge is to determine which of the Invoices sent to a problematic customer

have been settled by a sequence of Payments. Your ReconcilePayments routine needs to examine theInvoices and thePayments input arrays and produce an array of Reconciliation records. Each Reconciliation record must contain the paymentNumber of the Payment being matched, the invoiceNumber of the Invoice being matched, and the amount of the Payment being applied to the Invoice. A given paymentNumber may appear in multiple Reconciliation records, if the Payment reimbursed more than one Invoice. A given invoiceNumber may also appear in multiple Reconciliation records, if partial payment was provided by more than one Payment record.

There are a few things that you can rely on during your reconciliation. Customers never pay in advance – the payment date is never earlier than the date(s) of the Invoice(s) being paid. Customers might make a partial payment, but when they do so they never combine the partial payment with payment for any other Invoice.

Payments might be matched to Invoices in a number of ways, so more than one solution might exist. We have some constraints, however, that reduce the amount of ambiguity. Since we are going to use your results to charge interest to our problem customer, and we want to be as fair as possible, you must apply Payments to the earliest applicable Invoice. If the Payment matches a single Invoice exactly, or matches the sum of a number of Invoices exactly, you must apply the Payment to that (those) Invoice(s). If a Payment exactly matches the balance of an Invoice that has been partially paid by a previous Payment, you must apply the Payment to the balance of that Invoice, unless some other rule applies (e.g., an earlier Invoice that is exactly paid by this Payment).

To support assessing an interest charge for our problematic customers, you need to help the accounting department by calculating how much this customer is in arrears. Specifically, you need to calculate the number of dollar-days the customer is overdue. This is the sum, for each unpaid or partially unpaid Invoice, of the unpaid balance of the Invoice times the number of days the Payment is late. The unpaid amount is simply the amount remaining after your Reconciliation calculations, and the number of days late is the difference between the date of

the most recent Payment and the date of the relevant Invoice. The sum of the dollar-days these Payments are late must be returned in the lateDollarDays parameter of the ReconcilePayments call.

Remember that you cannot apply a Payment to more than one Invoice unless the Payment exactly totals the original amount of those Invoices – you cannot simply apply partial Payments to the oldest Invoices to minimize the lateDollarDays value.

The winner will be the solution that correctly calculates the lowest lateDollarDays result. Among tie values, the entry with the lowest execution time will be the winner.

This will be a native PowerPC Challenge, using the CodeWarrior Pro 5 environment. Solutions may be coded in C, C++, or Pascal.

This Challenge was suggested by Ernst Munter, who adds two Challenge points to his lead for the suggestion. Thanks, Ernst!

One final point. I've received some letters from people who would like to participate in the Challenge, but who find the problems too difficult (despite my best efforts to simplify them), or who find our veteran contestants to be too dominating. I'd really like to find a way that new contestants could feel comfortable playing, so I'm considering ways to revise the prize structure or the points system to make that possible, while still being fair to our regular contestants. If you have any thoughts along this line, please drop me a suggestion at progchallenge@mactech.com.

THREE MONTHS AGO WINNER

Congratulations to ... – oops, there is no one whom we can congratulate for winning the July RAID 5+ Challenge. The problem, you might recall, was to design a disk input-output system that would survive the loss of two disks in the array. A number of folks wrote to say that the problem was impossible, which must come as a surprise to the vendors who marketed solutions to this problem. In any case, this is one of those rare months when no one submitted a solution.

Is the problem impossible? Imagine a sequence of disks A, B, C, D, and E. Further, imagine that those disks are striped as follows:

```
A0 B0 C0 D0 E0
A1 B1 C1 D1 E1
A2 B2 C2 D2 E2
A3 B3 C3 D3 E3
A4 B4 C4 D4 E4
```


Now, according to the problem statement, the actual data capacity of our array only requires N-2 of the available N disks. So, to implement protection against the loss of one disk, we can imagine calculating parity for block (stripe) n across disks A, B, C, D, and E. Call those parity blocks P1, P2, ... Then imagine calculating another parity block, one for each disk. Call those parity blocks PA, PB, ... PE. Then lay those parity blocks out something like this:

```
PB B0 C0 D0 P0
A1 B1 C1 P1 PA
A2 B2 P2 PE E2
A3 P3 PD D3 E3
P4 PC C4 D4 E4
```

Notice that the parity blocks are rotated across the available disks, but that the parity requires the capacity of only two disks. Note also that there is the equivalent of 3 disks of information distributed across the 5 physical disks.

So what happens when two disks fail? How do we recover a block? Let's say that disks A and B fail, and that we want to recover block A1. (Failing these two disks actually represents a worst case, since the parity block PB for the failed disk B is on the other failed disk.) We have ten missing blocks of information (6 data blocks and 4 parity blocks), and we have the following parity equations, with the known elements bolded:

```
1:  B0 + C0 + D0 + P0 = 0
2:  A1 + B1 + C1 + P1 = 0
3:  A2 + B2 + E2 + P2 = 0
4:  A3 + D3 + E3 + P3 = 0
5:  C4 + D4 + E4 + P4 = 0
6:  A1 + A2 + A3 + PA = 0
7:  B0 + B1 + B2 + PB = 0
8:  C0 + C1 + C4 + PC = 0
9:  D0 + D3 + D4 + PD = 0
10: E2 + E3 + E4 + PE = 0
```





SPOTLIGHT

seven times faster

free demo
www.onyx-tech.com

Find memory errors automatically in source
 Code Fragment Support
 Leak Detection
 Toolbox Parameter Checking

FireWire with a DIFFERENCE

VST's line of portable USB, FireWire® and PowerBook® peripherals represent the ultimate in performance, ultracompact light-weight packaging, and plug 'n' play ease of use.

Taking FireWire beyond the limits

VST's New FireWire RAID Array packs 120GB into the smallest possible space, supports RAID levels 0 and 1 simultaneously, and even operates on a PowerBook Li-Ion battery for up to three hours.

VST's New FireWire RAID Array is a powerful tool for demanding professional applications, such as video and audio broadcasting, where large amounts of graphics, digital video and music must be captured and manipulated in real time for storage or editing. For performance and high capacity in FireWire storage VST FireRAID™ is the solution!

Visit us at www.vsttech.com to learn more about this and other products designed to simplify the digital lifestyle.



Special Offer:
Sound Jam MP Free
& Alladin Software
Included

Full Height FireWire Hard Drive
With sizes available up to 75GB, the Full Height FireWire Hard Drive is the ideal solution for those looking for high speed desktop storage.

FireWire/USB Hard Drive
Our combo drive includes FireWire & USB ports in the world's smallest and fastest portable drive up to 30GB.

FireWire Hard Drive
VST's amazing, high performance FireWire hard drive with capacities from 3GB to 30GB.

ATARAID PCI Card
Add on to your RAID solution to your G4 or G5 with the ATARAID PCI card. Also available as a flexible unit up to 150GB of storage.

SMARTDISK
Simplifying The Digital Lifestyle™



The first equation allows one to determine B0, the second A1+B1, the third A2+B2, the fourth A3+P3, the fifth P4, and the eighth PC. The final two equations are all known quantities and do not help. This leaves us with the following:

$$\begin{aligned} 1: & B0 + C0 + D0 + P0 = 0 \\ 2: & A1 + B1 + C1 + P1 = 0 \text{ (A1 + B1 known)} \\ 3: & A2 + B2 + E2 + P2 = 0 \text{ (A2 + B2 known)} \\ 4: & A3 + D3 + E3 + P3 = 0 \text{ (A3 + P3 known)} \\ 5: & C4 + D4 + E4 + P4 = 0 \\ 6: & A1 + A2 + A3 + PA = 0 \text{ (A1 + A2 + A3 known)} \\ 7: & B0 + B1 + B2 + PB = 0 \text{ (B1 + B2 + PB known)} \\ 8: & C0 + C1 + C4 + PC = 0 \\ 9: & D0 + D3 + D4 + PD = 0 \\ 10: & E2 + E3 + E4 + PE = 0 \end{aligned}$$

This gives us 5 remaining equations and 7 unknowns (A1, A2, A3, B1, B3, P3, and PB). So, it looks like we are short a couple of equations to arrive at a unique solution. But we've ignored an additional constraint imposed by parity, that $x+x=0$. By enumerating all possible values of these seven remaining unknowns, and determining which constraints are satisfied by each combination, it turns out that only one combination of values satisfies all constraints. So it seems to me that a solution ought to be possible. If you think I've missed something, please let me know and I'll continue the discussion!

There is also a fair amount of literature on RAID schemes beyond RAID 5. If you're interested, you might start out at <http://www.pdl.cs.cmu.edu/RAID/RAID.html> and take it from there.

TOP CONTESTANTS

Listed here are the Top Contestants for the Programmer's Challenge, including everyone who has accumulated 10 or more points during the past two years. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

Rank	Name	Points	Rank	Name	Points
1.	Munter, Ernst	243	9.	Downs, Andrew	12
2.	Saxton, Tom	106	10.	Jones, Dennis	12
3.	Maurer, Sebastian	78	11.	Day, Mark	10
4.	Ricken, Willeke	65	12.	Duga, Brady	10
5.	Boring, Randy	50	13.	Fazekas, Miklos	10
6.	Shearer, Rob	47	14.	Murphy, ACC	10
7.	Taylor, Jonathan	26	15.	Selengut, Jared	10
8.	Brown, Pat	20	16.	Strout, Joe	10

There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place	20 points
2nd place	10 points
3rd place	7 points
4th place	4 points
5th place	2 points
finding bug	2 points
suggesting Challenge	2 points

BOOTCAMP FOR STARTUPS.SM THE FEW, THE PROUD, THE OBSESSED.



SILICON VALLEY, October 16-17

WASHINGTON DC, November 15-16

garage.com
we start up startups

Attention. Join Garage.com's two-day Bootcamp for Startups. Learn the fundamentals of taking your company from startup to IPO. Hear from the high tech industry's top investors, experts, and entrepreneurs. Gain invaluable information about raising capital, building a buzz, hiring top talent, and launching your product. At ease.

LOG ON TO **WWW.GARAGE.COM/BOOTCAMP** TO LEARN MORE & REGISTER TODAY.

Forbes

IBM

W&R

Wilson Sonsini Goodrich & Rosati

Microsoft

hp

invent

METRIUS

ADVANCED TECHNOLOGY

plural

tiVo

PRICEWATERHOUSECOOPERS

Silicon Valley Bank

BOWNE

REED SMITH

Morgan, Lewis & Bockius

STARTUPS.COM

mofo.com

ONYIA.com

CONEXION

netpreneur

San Jose Mercury News
The Newspaper of Silicon Valley

THE STANDARD

Silicon Valley Network

SiliconValley.com

spacedisk

by Jeff Clites <online@mactech.com>

CVS: Version Control is Your Friend

CVS, the Concurrent Versions System, is the standard in version control systems in the Unix world. Traditional Mac users may not have used CVS before, and in fact may not have used any version control system, but it can save you numerous headaches during your development projects, and with the arrival of Mac OS X there is no longer any excuse—CVS is free and may even come preinstalled with your developer tools.

So what is a version control system? In short, it allows you to manage and track changes to the files of a project. From an operational standpoint, most systems (including CVS) are based on a central repository which contains all of the files of your project, and individual developers “check out” copies of these files, make changes to their local copies, and then “check in” their changes to the central repository. In the process, the system does three basic things for you: it helps coordinate a team of developers working on the same project (so that programmers don’t interfere with other programmers working on the same files); it tracks what changes were made to what file, when they were made, why they were made, and who made them (this helps you identify the cause of a newly introduced bug and helps enforce developer accountability); and it allows you to revert to a version of your project that existed at some previous point in time (this allows you to test compatibility with previous versions or to undo destabilizing changes). Version control isn’t a magic bullet or a substitute for communication between developers, but it does go a long way toward preventing common (and annoying) problems. Version control is a must when multiple developers are involved in a project, but even a single developer working alone will benefit.

The main advantage that CVS has over most if its competitors is that it uses optimistic locking rather than pessimistic locking for checked-out file. With pessimistic locking, only one developer can check out and modify a given file at one time, in order to prevent two users from stepping on each others changes in the same file. This sounds like a good idea on the surface, but in practice it can become a major impediment if users keep files checked out for long periods of time, as they would

naturally do when editing them, or as they may accidentally do if they forget to check them backing in before leaving for lunch, for the day, or for vacation. Ultimately, users will resort to circumventing the system to obtain a copy of a “locked” file so that they can get some work done, and they’ll have to manually reconcile their changes with the user who originally checked out the file. CVS takes a completely different approach—it doesn’t lock access to files at all. Multiple users are free to check out and work on the same files at the same time, and the system takes care of merging changes together as users commit their modifications back into the central repository. The majority of the time, users don’t actually end up editing the same files at the same time (even though they may have checked them out, for instance to look them over for needed changes), and even when they do CVS is able to merge these changes together automatically. In the rare case in which two users have both modified the same line of code between check-ins, CVS notifies the user of a conflict so that he can resolve it manually. This sound like it could cause a mess, but in practice it usually does just what you would have wanted, and it only makes you spend your time resolving the rare conflict when it actually does occur, rather than causing ongoing interruptions in order to ward off potential problems that never actually arise.

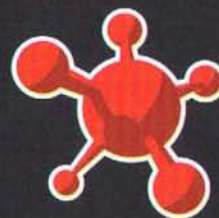
CVS has many strong points in addition to the above-mentioned “non-locking” behavior. For instance, it is open source and licensed under the GPL, and therefore is available free of charge. Also, despite its Unix heritage, there are GUI-based clients available for the classic Mac OS and for Windows. On the other hand, CVS is far from perfect—it has been said that it’s terrible but it’s the best thing around. Many of its flaws are merely nits, but they can add up to a noticeable inconvenience. To begin with, CVS shares a problem which is unfortunately common among open-source projects, namely that it has a significant amount of documentation yet it is often difficult to find the exact information you are looking for. For example, a very nice feature of CVS is that it can be configured to run over SSH, so that passwords and data are encrypted during transmission. It’s hard to find a complete



REAL PEOPLE, REAL SUPPORT.

MacTank provides technical support solutions for Macintosh application developers. We support your end users so you have time to concentrate on marketing and development. Bring your applications to OS X from Windows or NeXT and we will do the rest.

For pricing and information
call 877-751-2300, option 2.



MACTANK
THE TECH SUPPORT ALTERNATIVE

Visit us at www.mactank.com



discussion of this feature, and the setup is such that it's easy to think that you've enabled this feature when in fact you haven't. (There also doesn't appear to be a way to force the use of SSH from the server side, i.e. to ensure that clients cannot connect insecurely.)

A second annoyance is the somewhat inconvenient handling of directories. CVS fundamentally works on a per-directory basis, and in each directory it places a "CVS" subdirectory with configuration information. This can cause problems if you duplicate a directory and forget to remove the duplicate CVS information. Under Mac OS X there are many types of "files" which are actually directories, and tools which manipulate these need to be CVS-aware so that they don't remove the CVS information when saving. Some of these "magic directories" (for instance InterfaceBuilder's .nib files) are best thought of as opaque binary files, whose contents should not be manipulated by merge tools, and there is a "wrappers" facility which allows you to treat them as such but, inexplicably, this feature is not compatible with client-server operation. Fortunately, Codefab maintains a modified version of CVS which does support both of these features, although it's a shame that the CVS maintainers don't seem to be interested in merging this back into the core distribution. This still doesn't solve the problem for "magic directories" which you don't want to treat as binary (such as EOModeler's .eomodeld files), but fortunately the latest incarnations of Apple's developer tools seem to handle the CVS directories correctly. For older version, the Omni Group has plug-ins which fix this problem, for both EOModeler and InterfaceBuilder. The directory-centric nature of CVS also has the unpleasant side-effect that you can't really remove a directory from your project — you can only remove the files which it contains. There is a workaround in that there is a setting which tells CVS not to create empty directories during a checkout, but this has the side effect that you can't place an empty directory under version control. (It's rare that you would need to, but it's important to keep in mind in case you ever do.) On a related note, you can place binary files (such as images) under version control, but you have to go through a bit of special configuration to flag them as binary so that CVS doesn't corrupt them, for instance by doing line-ending conversion. Often, these types of files do not really need to be under version-control per se, but it's very convenient to place them in your repository nonetheless so that you can maintain a complete copy of your project, especially in web development projects which tend to contain a large number of image files.

Codefab CVS with Wrapper and Client/Server Support

<<http://www.codefab.com/cvs.html>>

Omni Group Plug-ins

<<http://www.omnigroup.com/community/developer/eomodeler/>>

<<http://www.omnigroup.com/community/developer/interfacebuilder/>>

DOCUMENTATION

Apple itself uses CVS for some of its internal development projects, and in fact Mac OS X Server shipped with it pre-installed. In particular it is used with the Darwin project, and there is a page on their Public Source site which serves as a good jumping-off point to find documentation and other information about CVS. Here you'll find a brief introduction to CVS concepts, by Apple's Wilfredo Sánchez. A good next stop is Jim Blandy's "Introduction to CVS", which goes into a bit more detail. Next up is "Open Source Development with CVS", a book by Karl Fogel. Generously, although this is a commercial book, the chapters on CVS itself are available for free in a variety of electronic formats. It is extensive, well-written, and a pleasant read. Finally, there is the core CVS manual, "Version Management with CVS", colloquially referred to as "the Cederqvist", for the name of its original author. It is large and complete, but it may not be the best place to start out. Finally, for an account of using CVS specifically with web development projects, take a look at "CVS Version Control for Web Site Projects".

Apple - Open Source - CVS - Docs

<<http://www.publicsource.apple.com/tools/cvs/docs.html>>

CVS Concepts

<<http://www.publicsource.apple.com/tools/cvs/concepts/>>

Introduction to CVS

<<http://www.cvshome.org/docs/blandy.html>>

Open Source Development with CVS

<<http://cvsbook.red-bean.com/>>

CVS Manual: Version Management with CVS

<<http://www.cvshome.org/docs/manual/index.html>>

CVS Version Control for Web Site Projects

<<http://durak.org/cvswebsites/>>

Much of the core documentation is located on the main CVS site, CVShome.org, and it's a good place to look for additional information and to download CVS itself, despite the somewhat inconvenient organization of the site. Their documentation page has links to the most important references, including overviews of CVS and of version control, and a CVS commands quick reference. (O'Reilly has also just published their CVS Pocket Reference, if you'd like something a little more extensive and in printed form.) Also check out the CVS Bubbles site, and especially its documentation page, which has links to a number of tutorials and to other CVS sites.

CVS Home - Documentation

<<http://www.cvshome.org/docs/index.html>>

CVS Overviews

<<http://www.cvshome.org/docs/overview.html>>

<<http://www.cvshome.org/docs/version.html>>

CVS Quick Reference

<<http://www.cvshome.org/docs/ref.html>>

CVS Pocket Reference

<<http://www.oreilly.com/catalog/cvspr/>>

CVS Bubbles

<<http://www.loria.fr/~molli/cvs-index.html>>

ALTERNATIVES

If you're a classic Mac OS user, you'll have to run the CVS server on a Unix machine, but there are several different clients that you can run: MacCVS Pro, MacCvs, and MacCVSClient. Despite their similar names, these are separate applications. The first of these, MacCVS Pro, is sponsored by the Mozilla project, and appears to be the most current version.

MacCVS Pro

<<http://www.maccvs.org/>>

MacCvs

<<http://www.wincvs.org/>>

MacCVSClient

<<http://www.glink.net.hk/~jb/MacCVSClient/>>

Under Mac OS X, CVS should work "out of the box" thanks to the BSD layer. If you don't like the command-line version (which really is quite easy to use), there is a GUI front end, CVL (Concurrent Versions Librarian) by Sen:te. CVL works on top of the standard CVS rather than replacing it. This is nice, because it should be forward-compatible, and it also means that you can resort to the

command-line version if there is some additional feature you need to use which CVL doesn't support, and you can do this without worrying that you might "mess up" any of the locally-stored administrative information.

Concurrent Versions Librarian

<<http://www.sente.ch/software/cvl/>>

Finally, if CVS isn't to your liking after you've tried it out, there are several alternatives, and you might want to use one of the commercial systems. For the classic Mac OS there is VOODOO (Versions of Outdated Documents Organized Orthogonally). If you need support for multiple platforms, take a look at Perforce, which seems to be well liked and is available for a wide range of operating systems, although there don't appear to be any GUI-based clients for it. (While visiting Perforce's site you might also want to check out JAM, which is a build system designed as an alternative to make, and which is used as the build system under Mac OS X's ProjectBuilder.)

VOODOO

<<http://www.unisoftwareplus.com/products/products.html>>

Perforce

<<http://www.perforce.com/>>

Jam - Make(1) Redux

<<http://www.perforce.com/jam/jam.html>>

MT



C++ Developer's Kit

PROFESSIONAL

Advanced Tools for ANSI C++ Development

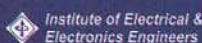
Suite of tools and reference collection showing developers how to create better ISO/IEC C++ code using UML&C++. It includes libraries of useful programming snippets and classes that you can use in your own projects to create more efficient powerful and reliable Mac applications.



From C to C++ more effectively!

C++ Developer's Kit Professional is a comprehensive C++ development kit for developers who need powerful C++ and UML features. This new development kit includes ISO/IEC C++ based examples for CodeWarrior, MPW, Symantec C++. This kit offers examples, projects, classes, functions libraries, templates, algorithms, tools and much more. It also provides over 250,000 code lines, the complete CAD++ library with classes and functions to design CAD applications, Garbage Collection-based classes, nested classes, UML/C++ models and more!

"C++ Developer's Kit offers a complete set of tools to build software conforms to ANSI/ISO C++ standards. The package also helps solve problems that might emerge in development. The software explains the fundamentals of UML, and OO development with a variety of models and applications."



COMPUTER

developer's
C++news
Technical News for Software Developers

Try out free C++ sample code at: <http://www.technosoftweb.com/C++>

CodeWarrior
LITE version INSIDE

TECHNOFT

<mailto:orders@technosoftweb.com>

www.technosoftweb.com



FREE shipping
(For Europe-CEE, USA and CANADA)

Just \$199 from
DEPOT
101 Avenue de la Gare 970
Tel: 01 47 36 00 00
www.depot.com

By Stephen Fantl

Copyleft or Copyright

Into the new paradigm

The world of publishing is about to change in a very dramatic fashion.

Publishers and distributors are about to find that the proverbial cat that has been hiding in the WWW has just escaped its bag. The form it takes is called the "Free Digital Library" (FDL).

Free at the FDL does not necessarily mean without cost, it defines free in the sense of "Freedom". For example, the open software movement gives to humanity their source code, the code is free as opposed to being an intellectual property right belonging to a single entity. This means the information is allowed to evolve and be shared for the benefit of all humanity. This concept of information being the intellectual property of all humanity can be termed "Copyleft".

The legal mechanism called "copyleft" assures:

- The author will be perpetually acknowledged in any use of her or his creative Works
- The Works will not become private property
- The Works remain available to inspire and be built upon by other creative minds

The creators of the FDL have two simple goals:

1. Attract and expand a knowledge base of Free Information;

The people who created the FDL make very clear a few points about their creation:

1. The project has no owners, stockholders, officers, or managers — only volunteers, donors, participants, and users.
2. It is "owned" by its users and exists solely in service to them.
3. It provides zero-cost services and "owns" absolutely nothing.
4. It was built, will change, and be operated entirely through volunteered time and effort, donated services, and financial support (if necessary).

5. It has been scrupulously constructed as a "human experiment" with no restraints on direction, speed, or content. It will "go" wherever and "carry" whatever Free Information is determined by its users.

Here's how it works. Creators of digital information use the FDL library of links to place their link to their information, providing the ability to freely self-publish. If Creators of digital work wish to commercially distribute their product for commercial compensation then they go through a different process.

The Commercial Distribution and Compensation process is one of the most unique aspects to what the FDL represents. At first glance the compensation plan looks akin to a chain letter, the differences being built in authentication, tracking ability, and verification controls for the distribution process. For example, if someone orders the offered product and does not pay you or information you ordered from a distributor does not arrive, then you would report them on the appropriate Peer Review site. If someone is selling information that is valueless (at least in your opinion) then you can review their work on the Authors Peer Review page. Since you have many "letters" to choose from that you might like to distribute, it's up to you to choose what you think is worthwhile. If you think none of the current Distribution Notices is valuable then provide your own creative works that can be distributed.

The Distribution Notice offers four articles by four distributor sources. Digitally self-published information is attached at each of the four levels that are sold for five dollars each. The author and the article never change positions, only the current distributors change when a new person chooses to participate because they will put their name and payment instructions in position one while moving everyone else up one level. Each work is \$5.00 so your total cost to own and redistribute all four products is a one time \$20.00 outlay.

The sources and titles will look something like this when you receive the e-mail offering the products and your opportunity to become a distributor:

Source 1 Position 1: Mary x – Article in position 1 title and Author Payment details (Paypal account or address)

Stephen Fantl is the Executive Director of S.A.F.E., a sustainable farming and organic agricultural Non-profit working on sustainable community models. Funding for the S.A.F.E. project will come from distribution by volunteers of their creative works through the FDL.

Works with
Apple AirPort



SkyLINE™ 11^{mb}

Join the wireless revolution.

802.11b for Mac OS
and Windows

*Wireless LAN access to email,
Internet, printers & more!*



 **Farallon**[®]
www.farallon.com

Download Free Wireless White Paper Today!

or Call (800) 613-4954



Source 2 Position 2: Bobby x – Article in position 2 title and Author
Payment details (Paypal account or address)
Source 3 Position 3: Jane x – Article in position 3 title and Author
Payment details (Paypal account or address)
Source 4 Position 4: David x – Article in position 4 title and Author
Payment details (Paypal account or address)

Here is a Hypothetical situation that shows how the process works:

1. Source '1' sends e-mail introducing the copylefted articles for sale and distribution to you.
2. Upon recognizing the apparent value of the Works offered, you decide to order all four and send a \$5 payment to each Source listed on the notice (the four authors or distributors through whom the notice has passed). Additionally, you must reply to Source 1 to request the Delivery Instructions for the entire set of Works.
3. Upon receiving your e-mail request Source 1 responds to your e-mail address with the Delivery Instructions for the set of Works (shown as Works 1, 2, 3, and 4). By taking ownership of the set of Works, you have agreed to make the \$5 payments to each of the four Sources (the four Sources listed on the Notice to which you are responding) in exchange for their efforts in the creation and distribution of the set of Works.
4. After initiating payment and following the Delivery Instructions, you now own the four copylefted Works. You resell in the same way in which they came to you (becoming yourself a new Source 1). By applying the same methodology in which you received the Works, you e-mail the same Delivery Instructions you received, in Step 3, to the others who responded to your Notice.

In a nutshell, by purchasing, thus owning copylefted Works, you have also received permission to sell them to others in the same manner you purchased them should you choose to do so. This model when coupled with verification and authenticity controls is a very effective and equitable tool for remunerating authors and participants who are interested in making money distributing Free Information. Even if you are not the author you can purchase a notice from someone like myself, pay a total of twenty dollars, five paid out to each four distributors, and then you would be returned an appropriate sum based on your efforts. Look at the math of this logarithmic distribution system.

Let's assume for our first example that between everyone there is an average of ten people per person with whom they distribute the information to who will do the same.

1st level: your 10 members each pay you and the other three people on the list \$5.00

$$(\$5 \times 10) = \$50$$

2nd level: 10 members from each of those 10

$$(\$5 \times 100) = \$500$$

3rd level: 10 members from each of those 100

$$(\$5 \times 1,000) = \$5,000$$

4th level: 10 members from each of those 1,000

$$(\$5 \times 10,000) = \$50,000$$

Remember that this assumes the people who participate recruit 10 people each. Consider what happens if they got 20 people to participate! It is common in the MLM industry for people to have hundreds of participants in their personal networks. However, because the width of your network is entirely self-determined, let's look at other network widths of 2 and 6 progressions.

2-person progression: 1st Level = \$10 - 2nd Level = \$20 - 3rd Level = \$40 - 4th Level = \$80 for a total of \$150.00

6-person progression: 1st Level = \$30 - 2nd Level = \$180 - 3rd Level = \$1,080 - 4th Level = \$6,480 for a total of \$7,770

The potential that this type of distribution and compensation package offers is truly staggering. It is a Win/Win/Win scenario for the Authors, distributors, and even the public as a whole since knowledge and information are being freely (Copylefted) circulated. Because the information you can purchase and resell is dipped from the endless stream of creative input flowing into the Free Digital Library, you can do it all again with entirely fresh information. That is, you can promote new information to your first generation Referral Group who originally purchased from you the first time round; as well as to any additional participants you attract into your organization, and each of them can repeat exactly the same thing with their respective first generation Referral Groups.

Do you see the power and potential once you have your first generation? When you get a new opportunity you will simply hit one button to offer it to everyone who made money with you previously. If someone you know did not succeed, then buy a notice from them and send it to your network, the person you bought it from (who didn't collect much) will now be assured to collect almost as much as you. How much you are making is up to you. To help with that I have links at the bottom of the page that will take you to places you can advertise on the Internet for free, follow these links and you should do phenomenal.

Although the Free Digital Library may be the first to employ this method of controlled logarithmic marketing, it certainly will not be the last. The site will be active by August 18, 2000 and will have at least two notices available for distribution. Implementing these distribution & publishing concept signals the beginning of the end for competitive publishing and distribution practices. Remember that you can publish digital art, digital music, software, creative writings, and even digital video through the FDL and begin to reap the economic benefits of your creative and energetic output. And why shouldn't you? I welcome your feedback.

If you find this concept interesting, you can visit the FDL at <<http://www.freedigitallibrary.com>>, <<http://www.freedigitallibrary.net>>, or <<http://www.freedigitallibrary.org>>. If you have questions or you would like to receive the first and most immediately available distribution notice, which is only sent out by individuals, never by the FDL itself, please e-mail: <<mailto:freedigitalinfo@lycos.com>> with the Subject: "Distribute".

Have fun publishing!

MT

Compare the best Mac products at the best prices.



MacBuy.com, The Macworld PriceFinder, reviews and compares out-the-door prices of available Macintosh products from a wide array of online vendors. You'll quickly find the best product for your needs, and you'll also find the best price. Shop and compare computers, new and upgrade software, printers and more – it's all there.

The Macworld PriceFinder is your comprehensive resource for choosing the best Macintosh products, getting the lowest prices and buying with confidence from the vendor of your choice.

MACBUY.COM

The Macworld PriceFinder
www.macbuy.com

©2000 Mac Publishing, L.L.C. Macworld.com are trademarks of Mac Publishing. Macintosh is a registered trademark of Apple Computer, Inc.

By Ilene Hoffman, Contributing Editor

Cable Modem Guide

WELCOME

Cable modem access has finally reached mass proportions. If you don't have it, you will soon. High speed Internet access is almost required these days to access all the Flash, Shockwave, QuickTime, and other useless crap posted by many web sites. The following notes are meant to help you wade through the cable abyss, so that you are not caught unawares.

1. First, find out which company is the cable tv provider in your town — because that's about your only choice. These companies have a monopoly in any town they cover.

2. If you happen to be one of the very lucky few who has more than one cable provider — Call their tech support line, or billing, or almost any number associated with the cable company — listen carefully to their hold music. As most of your communication with your cable provider will be waiting on hold you might as well pick a company who plays music you like.

(Recently my ears were assaulted with classical music, REALLY loud classical music from Mediaone, now Roadrunner, now AT & T. When I called RCN though, I was greeted with "Hold on I'm Coming" by the Temptations(?) and "Wait a Minute Mr Postman" by the Beatles — obviously RCN has a better

sense of humor! I was expecting Stevie Wonder's Don't You Worry Bout a Thing next.)

3. If you are lucky enough to have more than one cable service provider, when you call, see what their wait time on the phone is — I guarantee you won't need a stop watch for this one. Pick the company that puts you on hold for less hours.

4. Mediaone (AT&T) advertises broad band service — what they really mean is when you buy a head set so you can stay on hold all day and still get something done; they mean buy one with a wide head strap, so you'll be more comfortable, while you wait ... and wait ... and wait...

5. If you ever really need tech support be prepared to be the victim. First, they will tell you, you're on a Macintosh, which of course they support; but then they tell you to hold the left mouse button, and ask if Windows is running. Second, they will blame your OS, doesn't matter what version you're running — its your fault!

6. Once they've established that you're on a Mac, and that the OS is ok, then they'll blame your old version of TCP/IP (which of course is the newest version) or Open Transport — doesn't matter if you've been working fine for 2 years with your set up - its still your fault or better yet, Apple's programmers. Now, the blame is slowing being spread to a whole slew of people!

7. After holding for a hour to get tech support, and rebooting your machines ten times and discovering that its all your fault, and your patience is running thin — you make them CHECK their stuff... and yes, only to find out their service is in fact down, not only in your town; but in the five contiguous towns too. They just hadn't figured that out yet. (See the support call at the end of this rant.)

Ilene Hoffman is a Contributing Editor at MacTech magazine. Sometimes she just can't take it anymore. She is also the Administrator at MacFixIt.com Forums and has worked with many community-based web sites. Comments on this article can be mailed to: ileneh@mac.com.

9. Now, that you're completely frustrated with your cable provider and they have a land lock on your town, you might decide to explore DSL. DSL provides a whole 'nother set of small nightmares — starting with their request you buy their modem. They want you to buy their modem as soon as you order the service, but you can't use that modem until the service is activated — which in the Boston area is a good month wait. So, its hurry up and wait, but pay first.

10. The best thing is once you are connected you will have speedy Internet access ... for about 10 minutes. Due to the phone lines in your area, your speed will be about half what is advertised, so you can expect slightly faster downloads than using that 2400 baud modem you have collecting dust on your floor.

Ah... life in the fast lane...

A RECENT SUMMARY OF A SUPPORT CALL I MADE

Dial the cable company.

Listen to the message that their options have changed and to choose my option carefully.

Type in my phone number, when requested and press 1.

Listen the message that their options have changed and to choose my option carefully.

Type in my phone number, when requested and press 1.

Listen the message that their options have changed and to choose my option carefully.

Type in my phone number, when requested and press 1.

Hang up.

Dial the cable company.

Listen the message that their options have changed and to choose my option carefully.

Type in my phone number, when requested and press 1.(repeat 3 more times)

Be put on hold.

A half hour later, after being told 15 times "We appreciate your patience, please continue holding, your call is important to us," a person answers the phone, and asks for your phone number.

Support:"What is your problem?"

Me: "My cable modem is not working."

Support:"Is this for your television set?"

Me: "No, I don't have a cable modem on my TV, it's for the Internet."

Support:"What is the problem with your cable modem?"

Me: "My modem is unstable"

Support:"Is it going to fall over? Is it on a desk?"

Me: (holding my breath) "No, my modem lights are blinking, and unstable."

Support:"What's your modem look like?"

Me: "Well, its this black rectangular box... — look I can't connect to the Net, can I speak with someone else?"

Support:"Oh, you need Internet access support."

Me: wait on hold while Internet access support is flown in from Paraguay.

Support:"Hello, can I have the phone number you are calling in from?"

Me: Give my phone number, address, social security number, mother's maiden name, and medical history of every dog I've ever owned.

Support:"What seems to be the problem?"

Me: "My modem lights are blinking, and unstable."

Support:"Have you tried accessing our help pages on the Internet?"

Me: (chewing on my keyboard...) "Um, no, I can't get ON the Internet."

Support:"Ok, first unplug the modem."

Me: Scrunch into a tiny little ball, squeeze under the desk and crawl 2 feet in the dark to reach my modem. Unplug modem, and hit my head while returning to the phone.

Support:"Ok, now we must wait." (Note: they have a egg-timer which they use to time the amount of time the modem is disconnected.)

Me: Wait.

Me: Wait. Two or so minutes later...

Support:"Ok, plug it back in."

Me: Repeat contortions listed above.

Support:"Ok, Now what's the modem doing?"

Me: "Blinking."

Support:"Ok, let me check your line. Oh, there's definitely a problem, I can't ping your modem."

Me: "Have you tried checking the area to see if the service is down?"

Support:"Oh, we'd know if there was a problem with the service in your area."

Me: "Might I suggest you double-check?"

Support:"Be right back."...

Support:"Oh, your node in your town is down, there's no service right now."

Me: DUH.....

Support:"Gee, thanks for calling, I don't know why we didn't know the service was down. We'll send someone out and get the problem taken care of. Usually we know when a service in a town is down."

(Note: this is the fourth time this year their service was out and they didn't know it.)

Four hours later my modem service was back up.

DISCLAIMER

Any resemblance to any services living or dead is purely coincidental. I only report em as I see em. Your mileage may vary.

USBStuff

1-88-USB USB US
-or- (1-888-728-7287)

WorldWide Distributors of USB
and FireWire Parts,
Peripherals and Accessories

Hey Developers:

<http://www.usbstuff.com/developers.html>

1-877-4 HOTWIRE
-or- 1-877-446-8947

FireWireStuff

List of Advertisers

4D	55
Active Concepts	41
AD Software	19
Aladdin Knowledge Systems	5
Aladdin Systems	65
Anthro Corporation	37
BeeHive	89
Belkin Components	7
Blue Dog	45
Blue World	17
Bowers Development	47
Catalog Stuff	103
Developer Depot	28-29
Digital Forest	53
Drive Savers	69
EVue	15
FairCom Corporation	9
Farallon	99
Garage.com	93
IDG	86-87
Intego	33
Mac Publishing	101
Mac Show	79
MacTank	95
Main Event	10
Mathmaesthetics	1
MDG	61
Metrowerks	IFC
Mindvision	81
Onyx Technologies	91
OpenBase International	83
Page Planet	13
Panda Wave	23
Paradigma	35
Pervasive	BC
RadGad.com	67
REAL Software	21
Register.com	85
Scientific Placement	31
SGI	49
Small Dog Electronics	25
Smith Micro	63
Sniderman	39
Stone Tablet	27
SuSE	IBC
Technosoft	97
TerraSoft	49
Thinking Home	43
True Basic	57
UNI SOFTWARE PLUS	11
VST Technologies	92
WebEX	51

List of Products

4D and WebSTAR • 4D	55
ADB Device • BeeHive	89
Application Service Provider • Blue Dog	45
AppMaker • Bowers Development	47
Boot Camp for Startups • Garage.com	93
BugLink • Panda Wave	23
CodeWarrior • Metrowerks	IFC
c-tree Plus • FairCom Corporation	9
Data Recovery • Drive Savers	69
Development Tools • Developer Depot	28-29
Domain Name Registration • Register.com	85
Electronic Equipment • Small Dog Electronics	25
eSellerate • Mindvision	81
Equipment • Anthro Corporation	37
FireWire Components • VST Technologies	92
Flat Panel Display • SGI	49
Funnel Web 3 • Active Concepts	41
Gifts and Gadgets • RadGad.com	67
Hardware • Belkin Components	7
Installer Maker • Aladdin Systems	65
Internet Service Provider • Digital Forest	53
Job Placement • Scientific Placement	31
Lasso Web Data Engine • Blueworld	17
Linux • SuSE	IBC
MacBuy.com • Mac Publishing	101
MacHasp USB • Aladdin Knowledge Systems	5
MacShow LIVE • Mac Show	79
MGI • Page Planet	13
MPEG 4 Multimedia Technology • EVue	15
Multi-Media Communication • WebEx	51
NetBarrier • Intego	33
OO File • AD Software	19
Professional Macintosh and Internet Development • Thinking Home	43
RAD Studio • OpenBase International	83
REALbasic • REALSoftware	21
Resorcerer • Mathmaesthetics	1
Scripter • Main Event	10
SkyLINE 11MB • Farallon	99
Software Protection • Sniderman	39
Spotlight • Onyx Technologies	91
Stone Table • Stone Table Publishing	27
Tango • Pervasive	BC
Tech Support Alternative • MacTank	95
Trade Show - Macworld SF 2001 • IDG	86-87
True Basic • TrueBasic	57
USB Stuff, FireWire Stuff • Catalog Stuff	103
Valentina • Paradigma	35
VOODOO Server • UNI SOFTWARE PLUS	11
Web Catalog • Smith Micro	63
WebServer 4D 3.2 • MDG Computer Services	61
Yellow Dog Linux • TerraSoft	49

Mac OS X Porting and Development Showcase

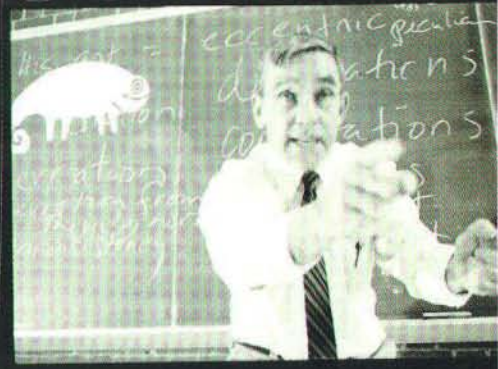
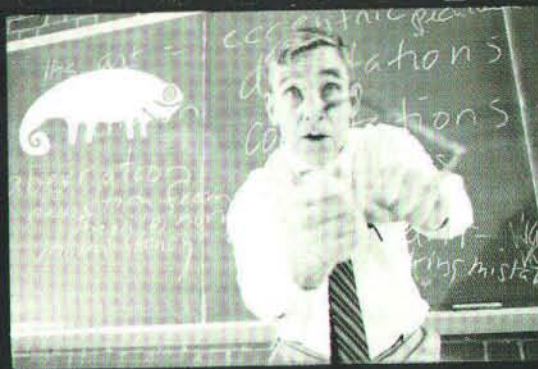
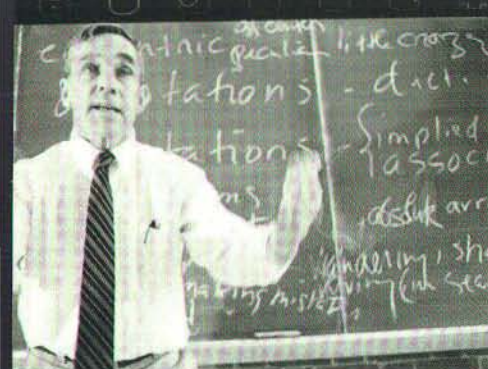
Art and Logic	75
Omni Group, The	74
Prosoft Engineering, Inc.	73
Red Rock Software	76
Robosoft	77
Shadetree	72

Mac OS X Porting and Development Showcase

Aqua Interface Implementation • Red Rock Software	76
Consultants • The Omni Group	74
Porting and Implementation • Robosoft	77
Programming Service • Prosoft Engineering, Inc.	73
Software Engineering Company • Art and Logic	75
Software Development Outsourcing • Shadetree	72

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

HOW DO YOU SAY SuPERB IN LINuX?



SuSE.

{soo' sah} is {soo' pûrb}

When you think of Superb, you think of unusually high quality. Like a superb wine, for example. Majestic. Rich. Luxurious. Superior. ■ SuSE Linux is all that. And more. More experience. More adaptability. More applications—over 1500. ■ More power to you. And more freedom, too. ■ No wonder more than 50,000 enterprises worldwide bank on its superb open source solutions. Making SuSE the international Linux leader—setting a higher standard for excellence, simplicity and support. ■ Even the price is superb. ■ So, how do you say superb in Linux? SuSE. It's a lesson well learned.

www.SuSE.com

Versions for Intel, Alpha, and PowerPC

The freedom to change.
The power to change the Linux world.





Time matters.

The fate of the company is in your hands.

Every second counts when you're competing at Internet speed. And the faster your Web application is developed, deployed and maintained, the faster your competition will drop out of sight.

The solution? Use Pervasive Software's Tango 2000 to develop your ideas with double the speed of any other development software.

- Visually develop applications on Mac, and deploy on Mac, Windows, Linux and Solaris
- Advanced XML support
- Connect directly to Filemaker or any ODBC database
- Extend applications with JavaBeans
- High performance Tango server scales with cluster support

Give yourself the same competitive edge that Tango has given leading companies like Apple Computer®, theglobe.com™ and Harbor Freight Tools™.

Download your Free Tango Demo today!

Grab the time-saving advantages of Tango 2000 absolutely FREE. Visit our Web site and download our FREE fully functional Tango 2000 demo. Or call us now to get your demo on CD.

www.pervasive.com/speed or 1-800-287-4383

Hurry! This offer ends soon.

PERVASIVE
SOFTWARE
The Freedom to Create Applications
for Everyone, Everywhere